2

STUDIES OF AN OPTICAL MULTI-PROCESSOR INTERCONNECT

US Army Grant Number 60-91-C-1043

Final Report

Principal Investigator

Jon R. Sauer

Optoelectronic Computing Systems Center

University of Colorado at Boulder

Boulder, CO-80309-0525

CONTENTS

TABLES

FIGURES

CHAPTER 1

AN OVERVIEW

1.1 Introduction

Optical fiber provide a very high capacity medium for point to point transmission of data. It is not straight forward to achieve the same level of success in networked fiber communication systems. A carefully designed system can reach terabyte-per-second capability. The purpose of this study was to investigate the performance of a generic optical distributed interconnect architecture for parallel processing. In particular, the target was the general purpose, easily programmable tightly coupled multiple-instruction stream, multiple data stream (MIMD) systems.

We have proposed a fiber-optic interconnect system that is aimed at exploiting the advantages provided by both optics and electronics to provide a low latency, high throughput communication between processors and memory modules in a shared memory system. Decisions taken during any new design are to be based on the basic requirements placed on the system, the strengths and weaknesses of the implementation domain as well as the identification of the problems that exist in past designs. Simulation is an invaluable technique for making design decisions with respect to new (non-existing) architectures. Hence, the primary goal of the study was to use simulations to aid the development of the proposed network through a realistic simulation of the teraflop processor architecture it is aimed to support.

Lack of efficient static storage in optics places the most important restriction on our architecture that the network be a flow through structure. In traditional packet-switched networks, all but one of the packets contending for an output link are stored to be transmitted later when the link is available. The hot-potato/deflection routing protocol used first in the HEP supercomputer does not need static storage at routing nodes. Also, the protocol is simple and can be implemented efficiently for fast packet-switching of short packets that are typical in shared memory multiprocessors.

A number of network topologies are suitable for deflection routing. In deflection routed networks packets utilize the network links as dynamic buffers as compared to dedicated static buffers in conventional store-and-forward networks. In simple terms, a deflected packet essentially takes a longer path to its destination from the point of deflection. So, they utilize network bandwidth that will be otherwise used for transmission of new packets. The average internode distance and the penalty for deflection are dependent upon the network topology.

1.2 Objectives of the proposal

With the above basic requirements in mind, we proposed to study a generic deflection routed interconnection network with different topologies. The original proposal was targeted at a two-pronged study of the advantages/disadvantages of the interconnect system. One prong was a simulation study of future large-scale parallel processing systems would perform if such an interconnect were developed. The other prong was a simulation study of the possible benefits to present supercomputer centers if some of their local and external interconnections were replaced by this interconnect architecture.

It was proposed to extend an existing simulator and use often used models of traffic as well as real traces to drive the simulations. It was proposed to employ two graduate students as research assistants to carry out the simulations under the guidance of the principal investigator(Jon Sauer) and other investigators (Harry Jordan and Andrew Pleszkun). The motivation for this proposal came from the Ultrafast Fiber Optic Network proposed by the principal investigator. A prototype demonstration of a multiwavelength node has been made under a companion project, mainly by then graduate student Daniel Blumenthal.

1.3 Ultrafast Networks

Ultrafast networks are characterized by the bandwidth differential between the sources and the network links. The memory access bandwidth sought by the processors is dependent on the processor architecture and the memory access latency. If only one request can be outstanding at any given time, the memory access bandwidth is the reciprocal of latency. Latency masking by multithreaded processors provides them with much higher memory bandwidth than suggested by the latency of an individual access. Assuming that the the processor can place one request per cycle, the limiting processor insertion bandwidth is 1. The access bandwidth available to the processors depends upon the network organization. Indirect networks (Fig. (1.1)) can provide an access bandwidth of 1 irrespective of the number of switching nodes in the system. The maximum access bandwidth in direct networks (Fig. (1.2)) is a function of the network size, the network topology and the degree of the network nodes.

Consider an N node direct network with switches of degree p and bidirectional links. Let each node be attached to a processor-memory pair. If $\langle E \rangle$ is the average number of internode links traversed by messages during a

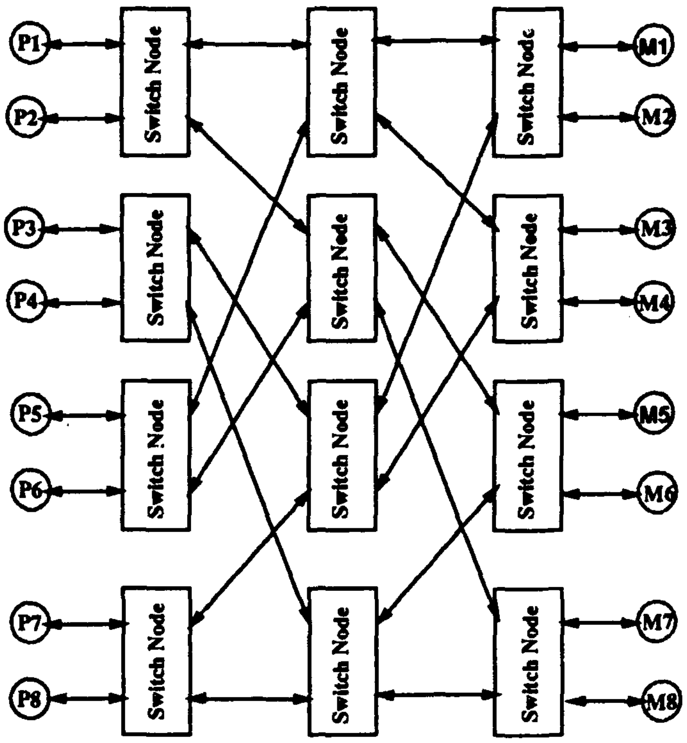


Figure 1.1. Organization of processors and memories in an indirectly connected multiprocessor

round trip, the maximum insertion bandwidth available per processor is

$$\gamma_{max} = \frac{p}{\langle E \rangle}. \tag{1.1}$$

The higher insertion bandwidth possible using an indirect organization is at the expense of an increase in number of switch nodes. The number of switch nodes in a directly connected multiprocessor is equal to the number of resources using the network. For a system with N processor-memory pairs, the number of switching nodes is N. The number of switch nodes in an indirectly connected multiprocessor depends on the size of the system as well as the topology of the interconnection. For example, a N processor, N memory system interconnected by 2×2 Omega Network has $(N/2) log_2 N$ switch nodes.

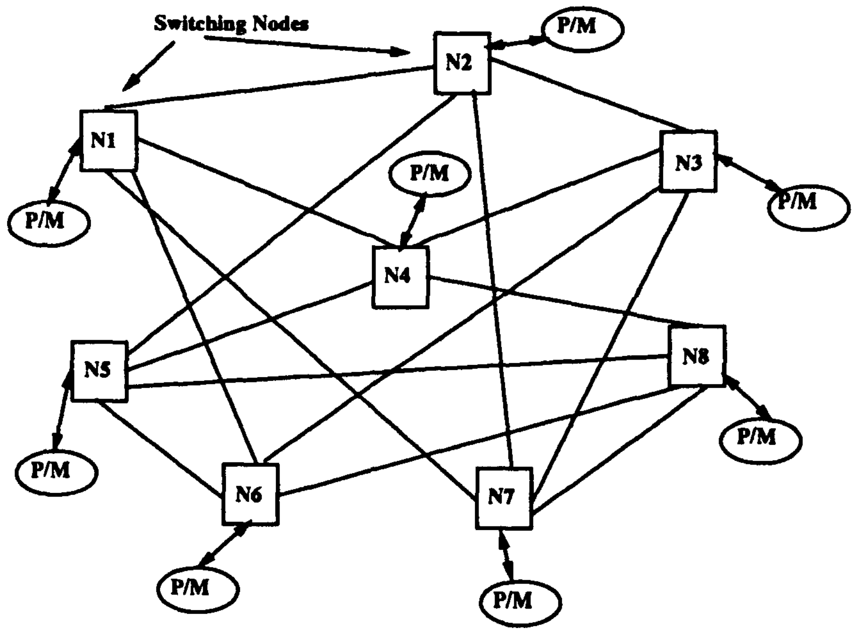


Figure 1.2. Organization of processors and memories in a directly connected multiprocessor

1.4 Principal Achievements

The simulation study was complimented where ever possible by analytical studies. Early simulations showed that it will be very hard to find a network topology that has acceptably low average internode distance and low penalty for deflection. It was recognized that other means for reducing the probability of deflection of packets need to be found to enhance the performance of the network. The conceptual pursuits led to the development of the idea of space-time switching, a technique that enhances the performance of the network to near ideal conditions under uniform loading. The benefits of space-time switching was predicted analytically and were verified through simulations. The concept seems to be patentable and hence an invention disclosure was made. No further action has been taken to file the actual patent application. Other significant contributions of the study are an accurate analytical model of the performance of a ShuffleNet, one of the two topologies investigated. The model and the simulator were developed concurrently so that one provided a validation tool for the other. A model multiprocessor system was simulated under synthetic loading conditions. The system was presented with spatially and temporally non-uniform traffic apart from an uniform load. All this study led to the PhD thesis [1] of one of the research assistants (Aruna Ramanan) employed on this project. A number of conference publications [2, 3, 4] resulted from the study. A paper has been submitted to a special issue of the Journal of High Speed Networks on optical networks [5]. A number of other papers are under preparation. Some interesting results were obtained using processor traces as work load to the simulator. The results have led to a definite thesis proposal as well as a technical report [6] by the other graduate student (Michael Sprenger) on the project.

This report presents the results of our study. The study is based on extensive simulations, which have been validated where ever possible by analytical techniques. A major outcome of this work is the identification of the potentials of simultaneous switching in space and time at network nodes. The report presents the proposed architecture, an analysis of the space-time node and the performance of the network obtained through simulations. With temporal reordering at nodes using the proposed architecture, the performance of the network gets very close to an ideal network when a uniform load is presented. Two network topologies were studied. The logarithmic ShuffleNet topology has been found to be more efficient than the Manhattan Street network, a two dimensional toroidal mesh. An accurate closed form analytical model of the ShuffleNet is presented. Different configurations of a model multiprocessor system, under both spatial and space-time switching and with the two topologies, were investigated. The behavior of the system was studied under uniform traffic, under spatially non-uniform traffic in the form of hot-spots, and under temporally non-uniform traffic in the form of bursts. The general conclusion is that it is possible to adjust the design space of a shared memory multiprocessor system to exploit the potentials offered by ultrafast networks. With appropriate support, from system and application software, such systems can utilize network resources efficiently and provide very high performance.

1.5 The Simulator

Two versions of the simulator were developed by the two graduate students. One was an extension of the original simulator structure aimed at extracting the network behavior in detail. The other was developed independently to have a detailed simulation of the host processors. The simulators are written in C++, an object oriented language that provides capabilities for information hiding. Both simulators run on DEC and Sun workstations. The first simulator was initially developed on a Sequent multiprocessor system as a uniprocessor program. The network simulator version has been optimized for fast simulation of up to about 2000 node systems. 1000 clock cycles of a 2000 node system with a short internode link (like the ones in existing systems) can be simulated in less than 15 minutes of system time. The efficiency is achieved through minimal simulation of the processing at nodes. The requests are generated using probability distributions, thus avoiding large storage for trace data. The second version is capable of simulating detailed processor behavior using real traces. It allows for variations in processor architecture such as the number of processes that can reside on the same physical processor. A common network network interface unit was designed and appropriate common data structures were developed so that the two simulators could be combined in future. For example, the first simulator could use the processor part of the second simulator. The only constraint will be the number of nodes it can handle. The simulators are generic and hence, can ported to supercomputers which can afford to provide large storage.

1.6 Overview of the Research

The focus of this work was on ultrafast networks for shared memory multiprocessors. It is possible to design packet switched direct networks with short word sized message packets in which the network link bandwidth is much greater than the insertion bandwidth of the sources. The choice of network architecture is critical in limiting the network's contribution to memory access latency as well as in exploiting the large available bandwidth.

Packet switched networks traditionally utilize store-and-forward techniques to resolve output port contentions by holding packets in an intermediate buffer until the required outgoing link is available. An alternative strategy is the hot-potato or deflection routing technique first proposed by Baran [7]. The main idea of the strategy is to minimize the necessary processing and storage overhead at each node by routing a packet out through an alternate output port rather than holding it until the desired output port is available. The basic requirement to implement deflection routing is the existence of paths from every node to every other node in the system. Packets will continuously move along in a flow through manner until they reach their destination. Instead of providing buffers at switch nodes the whole network, especially the transmission links, behaves like a dynamic buffer. The HEP multiprocessor system [8] was the first commercial system to use this strategy. The TERA [9] supercomputer is the latest massively parallel shared memory multiprocessor that is being developed based on this routing strategy. Recently, a lot of interest has surfaced on this routing method [10, 11, 12, 13, 14]. The main reason for this upsurge is that the deflection routing technique does not require asynchronous storage of through going packets and hence it is well suited for lightwave flow through architectures. While optics can provide enormous bandwidth, electronics has an edge over it in logic and memory technologies. Conversion of optical data into electronics at nodes for storage and reconversion to optics for transmission will necessarily slow the network.

Though the deflection routing protocol has many desirable features, a main drawback is the effect of the penalty for deflection. Deflections cause packets to reside in the network longer, the effect of which tends to be severe as the internode distance is increased. Deflection occurs when more than one packet arriving simultaneously requires the same output port. If packets can be rearranged temporally, then they can be reordered so as to reduce the contention for spatial channels. A time-slot interchanger for arbitrary temporal reordering of a sequence of N packets in flight can be accomplished using $O(log_2 N)$ switching elements [15]. Recently, a number of architectures have been proposed for multi-spatial channel time-slot interchangers [16, 17, 18]. In this work we focussed our attention on 2-space channel, 2-time channel (2S2T) switch nodes, though the concept is applicable to an arbitrary number of spatial as well as temporal channels. Our 2S2T switch nodes are simpler than a general 2S2T multichannel time-slot interchanger since we require only a subset of the 24 permutations possible by the latter. While traditional time-slot interchangers permute within nonoverlapping time frames, we shall consider a sliding time window. In our 2S2T switch, a contention for output port at a switch node can be resolved either by the preceding pair of packet slots or by the following pair of packet slots, if they have the capability to do so. Hence, a sliding time window provides greater contention resolution capability to the switch node than fixed time frames.

Deflection routing requires that the network be such that a deflected packet can reach its destination. Thus, the network topology should be such that any node in the network can be reached from any other node. A variety of topologies satisfy this requirement. Two widely different topologies are considered in this report. One is the ShuffleNet [19], a cylindrical network with logarithmic number of columns and the other is the Manhattan Street network (MSNet) [12], a toroidal mesh network. The ShuffleNet lends itself to tractable analysis. Though other researchers [20, 21, 22] have analyzed the ShuffleNet before, the model of the ShuffleNet presented here provides a simple closed form characterization of the network properties. The key feature of the model is the exploitation of the fact that at any time during its flight through the ShuffleNet, a packet will *care* about the output port assignment only during the last $min(l, k)$ hops. Here, l is the distance between the packet's current node and its destination, and k is the number of columns in the network. As will be shown, the MSNet is not amenable to such an analysis. The performance of both the ShuffleNet and the MSNet has been studied through simulations, thus providing a comparison between the behavior of the two topologies. The theoretical analysis and simulations of the ShuffleNet validate each other.

The probability of deflection of a packet at nodes along the way to its destination determines the residence time of the packet in the network. In a network, in which the request pattern of the sources is uniform both in space and time, probability of deflection of packets will be a constant throughout the network at all times. For any network with 2×2 nodes, this probability varies linearly as the link utilization as well as the care probability. The care probability is the probability that the packet will require a specific port at this node in order to reach its destination in minimum time. We have found that in our 2S2T node with sliding time window, the probability of deflection falls to approximately a cube of the product of the link utilization and the care probability. This reduction in the probability of deflection provides the network the capability to overcome the drawback of the equivalent spatial deflection routing network.

While the advantages of limited space-time switching in deflection routing networks have thus been predicted and verified under under uniform loading, a detailed study is needed to ascertain its suitability as a multiprocessor interconnect. To this end, the behavior of a model multiprocessor system under both uniform and non-uniform workloads has been studied as part of this research effort. The model system is a fully pipelined multiprocessor system with pipeline cycle times constant throughout the system. Behavior of the system under non-uniform workload is quantified through simulation study of spatial and temporal non-uniformities of processor request pattern. While spatial non-uniformity was studied through investigating the behavior under time-invariant hot spots, temporal non-uniformity was studied through investigating the behavior under spatially uniform temporal bursts. Limited temporal reordering at nodes provides the network the capacity to perform at near optimal conditions under uniform load. The ShuffleNet topology exploits the benefits of space-time switching, particularly in large systems. The hot-spot behavior of the system is not limited by the network, but is limited by the memory access bandwidth. Hot-spot being a spatial problem, it affects the purely spatial and space-time networks to the same extent. Since temporal bursts concern distribution of network accesses by the hosts in time, the space-time configuration adjusts far more quickly to bursts than the purely spatial configuration.

1.7 Organization of the Report

Chapter 2 presents the state of current knowledge on topics related to the research problem addressed by this work. An overview of shared memory multiprocessors is presented at the beginning of the chapter. It also presents the current trends in optical interconnects for multiprocessors and covers the work done by others on deflection routing. Chapter 3 presents an overview of the simulator. Chapters 4 through 6 present and analyze the work done as part of this research effort. Chapter 4 presents a simple model of the ShuffleNet topology under uniform load. This topology lends itself to an analysis that leads to a closed form relation between the network parameters. The predictions of the model have been verified by simulations. Also included in Chapter 4 is the qualitative analysis of the MSNet and the results of simulations under conditions similar to those under which the ShuffleNet was investigated. Chapter 5 presents a significant contribution of the work, i.e. Space-Time switching. A detailed derivation of the probability of deflection in a 2-space channel, 2-time channel switch is presented. The behavior predicted by the space-time model has been verified through simulations for both topologies. In both cases, the marked reduction in the probability of deflection leads to near ideal behavior up to moderately high uniform loads. Chapter 6 presents the multiprocessor model and results of the simulation study of the architecture underlying the model. The model covers various aspects such as the system description, the parameters of the system and their range, the workload characterization and the performance metrics of relevance. All these aspects put together form the basis for the simulation study and the results presented thereof. The last Chapter presents the summary and conclusions. This work addressed the main drawback of deflection routing networks and resulted in a new implementable solution that can help reduce the negative effects of deflection routing. The study has led to the identification of a number of key features of ultrafast multiprocessor networks operating under a deflection routing protocol. Directions for further work are presented at the end of Chapter 7.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Shared Memory Multiprocessors

Shared memory multiprocessors are particularly attractive for solving large scale scientific problems. The principal advantage of shared memory architectures is ease of programmability. The ideal multiprocessor, called a paracomputer by Schwartz [23], will be one in which access to shared memory can be achieved in one cycle. Though this ideal limit cannot be realized, the objective of any design is to increase the memory access bandwidth as much as possible. Memory access bandwidth and network latency are key parameters in determining the efficiency of tightly coupled multiprocessor systems. Hence, architectures that reduce latency and further increase the memory access bandwidth by masking the physically unavoidable components of processor/memory latency are particularly important for scalability of such systems.

The architecture of shared memory multiprocessors has changed drastically over the years. Until recently, directly connected distributed resources have been used primarily in message passing architectures in which processors communicate directly with each other. The direct connectivity among processors has been recognized as a key factor for scalability attributed to such systems. Such configurations were rarely considered for shared memory architectures because of their non-uniform memory access characteristics. Non-uniformities in memory access can be masked using latency hiding and latency reduction techniques. Whenever shared memory access is not uniform, each processor will perceive different latencies while accessing local and non-local memories. The term non-uniform memory access (NUMA) multiprocessors is used to refer to machines which support a shared address space in hardware, but have performance characteristics which vary significantly with the placement of the data. BBN Butterfly [24] was an example of a NUMA system. Another example is Cm* [25], which is a MIMD system based on hierarchical busses. It consists of a large number of computer modules grouped into clusters and connected by a hierarchy of busses. Cedar [26] is an hierarchical system with processors within a cluster connected by a crossbar with cluster memory they share and clusters connected to the global memory by a multistage interconnection network. Systems such as these are intermediate between systems in which all memory is global and external to the processors, providing uniform memory access, and systems in which memory is physically distributed among processors, leading to non-uniform memory access.

Distributed shared memory multiprocessor systems have memory associated with each processor. Each processor's local memory is either wholly or partially shared among all processors. The processors are connected by means of some interconnection network such that all nodes can be reached from all other nodes. A variant of this architecture is one in which processors and shared memory modules are separate, but are distributed throughout the network. Messages in shared memory multiprocessors are typically word sized. Also, path requirements of messages through any switching point change frequently. Hence, locally controlled, packet-switched networks are particularly suitable for distributed shared memory systems. The HEP multiprocessor system [8] is the first distributed shared memory multiprocessor. Currently, a number of distributed shared memory machines have been proposed, some of which are being built (e.g. [27], [9], [28]).

Latency associated with memory access has a profound influence on the processor performance in a shared memory system. Both interconnection network latency and memory interference contribute to memory access latency of these systems. As the system size grows, the interconnection network latency becomes predominant and tends to limit the minimum latency attainable. Often, the network latency is much larger than the communication time suggested by the network bandwidth. The effective latency can be reduced either through latency reduction or through latency hiding or both [27]. Caching is used in multiprocessors as a technique to reduce memory access latency by avoiding long latency operations [26, 29]. Latency hiding is achieved using latency tolerant processors that keep the processor busy while waiting for communication on behalf of some process [30].

Cache memories are traditionally used to improve system performance by minimizing the effective memory access time. Requirements for maintaining consistency between cache and main memory tend to be more complex for shared memory multiprocessors than for large uniprocessor systems or loosely coupled multiprocessor systems. For shared memory systems, the considerations vary depending on whether the communication system is bus based or based on multistage interconnections. A detailed analysis of cache requirements for tightly coupled multiprocessors is presented in [31].

Latency tolerance can be achieved using multithreaded processors.

The conventional single-stream or main-stream processors have a single process or task loaded at a time. When the process blocks, either during synchronization or due to remote access delays, the processor either remains idle until execution can resume or switches context to another ready to run process. Such context switches usually involve heavy overheads. In contrast, multistream processors have several active processes loaded at once and can switch from one to another very fast (HEP [8], Horizon [27], TERA [32], MASA [33], [34], [35],[36]).

The processors in the series of general purpose multiprocessors from HEP to TERA can context switch every cycle. Instructions are issued in succession from a ready to execute process resident on the processor. When an instruction completes, the stream to which it belongs thereby becomes ready to execute the next instruction. The basic idea is execution of multiple instructions in pipelined hardware rather than multiple processors. Hence the processor will be fully utilized as long as there are enough instruction streams in the processor, so that the average instruction latency is filled with instructions from other streams. This simple programming model of no instruction lookahead within a stream is being expanded to include overlap with compiler assisted explicit dependence lookahead in TERA. In MASA, a processor for parallel symbolic computing, any loaded task that is not suspended is eligible to issue the next instruction; instructions from different tasks can be issued on consecutive cycles. The fast context switching mechanism allows the processor to perform useful work on other tasks while a task is blocked.

An intermediate between the two extremes of single stream processors and multi-stream processors which switch context every cycle are coarse-grained multithreaded processors. In APRIL [37], a coarse-grained multithreaded processor, a single process continues to execute as in a single-stream processor until it blocks. The processor is provided with hardware mechanisms that facilitate fast context switch to another ready to run thread with just 4-10 cycles overhead. The hybrid data-flow/von Neumann machine described in [34] also employs coarse-grained multithreading. In the above multi-threaded architectures cited so far, the processor rapidly switches among threads. [38] proposes a processor architecture that can issue instructions from multiple threads simultaneously. The functional units are shared between multiple processors to form a united processor. The logical organization is as if there are still multiple processors. Multiple instructions from different threads are issued simultaneously and executed unless they conflict with one another for the same functional unit.

In all of the above descriptions the terms process, stream, task and thread have been used interchangeably. The number of threads that need to be interleaved on a processor to keep it busy will depend on the round trip latency of messages [39]. As the system grows in size, the the round trip latency will grow, thereby requiring more threads to be scheduled on a processor to keep the processor utilization up. A limit on the available number of threads that can be scheduled on a processor is placed by the applications running on the system. Hence it is desirable to reduce the round trip latency of messages to the extent possible so that latency tolerant processors could be used effectively.

A latency reduction mechanism that is more closely related to the network than the processors is the use of high speed networks. The M3S project [40] aims at achieving this by using a set of ultra high speed serial links. In the MIT Alewife machine [28] the network switches are clocked twice as fast as processors. The architecture utilizes a two-dimensional mesh interconnect with two unidirectional links in opposite directions between each pair of nodes. A study based on this machine model to assess the effect of communication locality on large-scale multiprocessor performance shows that the sensitivity to locality is lowered when the speed of the network is increased relative to that of the processor [41].

Apart from architectural aspects, there are a number of ways in which software, both system as well as application, can contribute to performance enhancement of a system. A typical example is the fact that it is possible to hide shared memory latency by relaxing the consistency requirements of memory access in cache coherent systems [42]. A combination of various hardware and software measures used to hide latency can be used to improve the performance of a system[43]. A detailed account of developments in areas such as compiler technology and algorithm development is beyond the scope of this work.

2.2 Role of Optics in Interconnects

Traditional electronic interconnects for large scale multiprocessors are severely limited by power dissipation and crosstalk due to resistance, capacitance, and inductance of the communication paths in the network. A typical example is presented by the Intel touchstone DELTA system which is limited to 512 processing elements [44] by the electronic interconnect. In order to scale the machine beyond this bottleneck it has been proposed to use a fiber optic mesh extender (FOME) to extend the routing mesh. The Galactica Net system uses a hierarchical interconnect, with the second-level interconnect being a 2-dimensional optical mesh [45]. An all optical polymer backplane for the Connection machine is presented in [46]. The turn to optics to overcome the fundamental constraints of electronics is justified because optics can provide a more efficient method for communication than electronics for distances greater than intrachip distances [47]. Optical interconnects offer the combination of large bandwidth and large fanout for a variety of computer applications [48]. The potentials and limitations of electronics and optics for interconnections at various levels are complimentary to each other in many respects. In the broadest sense, optics has the inherent advantages of speed, parallelism, reduced interference and crosstalk. An important factor that needs to be considered in the design of optical interconnects is that it is not possible to provide asynchronous memory. While bandwidth is cheap, other optical devices such as amplifiers, multiplexers, demultiplexers and switches are expensive resources.

A variety of optical methods can be used for realizing optical interconnects. While electronic switching systems can switch in space and time domains, optical systems can switch in wavelength-domain too, thus providing a greater range of switching techniques [49]. The basic requirement placed on the interconnect is to provide an efficient connection that satisfies the communication requirements of the electronic hosts. The functions to be carried out by an optical interconnection network are the electro-optic conversions at the sources and receivers, transmission, and switching. The transmission can be carried out either in free space or through waveguides. While guided wave transmission through optical fibers is suitable for larger distances, a suitable medium for lower level package to package, interchip and intrachip communications is either free space or integrated optic waveguides [50]. The large distance-bandwidth product of optical fibers allows larger separation between processors, thereby alleviating problems arising due to very close packaging of components of the system. Purely optical switching is currently not commercially practical. Photonic switching utilizes the combined strengths of optics and electronics, through electronically controlled switching of optical signals. A typical example is HYPASS [51], an optoelectronic hybrid packet-switched system, in which electronic components are used for memory and logic functions and optical components are used for routing and transport. A detailed account of architectural considerations for photonic switching networks is presented in [52].

Fiber optic communication has been used successfully for point to point communication in telecommunications and local data links. The usable bandwidth of fiber is about 30 THz [53]. Both circuit and packet-switched networks can be implemented using wavelength-division multiplexing (WDM) schemes that allow different transmissions at different wavelengths to be multiplexed concurrently on the same fiber [54]. HYPASS [51] is a wavelength division multiple access (WDMA) system with centralized passive star topology. The optical transport in HYPASS is based on a passive $N \times N$ transmissive star coupler that provides a single-stage crosspoint. The design is based on broadcast and select approach, with the star coupler combining light intensity from from every incoming fiber and distributing it uniformly to all output port receivers. The receivers select the wavelength assigned to carry information to them. The sources are provided with tunable lasers to transmit information on the wavelength of the desired destination. The main drawback of this scheme is that the implementation of the $N \times N$ star coupler does not allow cost efficient scaling in size. A multi-dimensional, k-ary n-cube based, architecture with one star coupled WDMA sub-channel for each set of nodes whose address differs in only one dimension is presented in [55]. Packets are forwarded from source to destination via intermediate nodes, with the packet undergoing optical/electrical (o/e) and electrical/optical(e/o) conversion at dimensional boundaries for routing purposes. Another approach is the multihop lightwave network which utilizes wavelength to effect the routing in a logical network that has a topology different from the physical broadcast star topology [56]. In the original multihop approach [10, 19] , packets are routed from the source node to the destination node via intermediate nodes in a sequence of *hops* on the fiber, each hop using a different fiber channel. The different channels are created in the fiber through assignment of a fixed number of wavelengths to the transmitters and receivers at each node, such that a logical connection is established between pairs of nodes that transmit and receive on the same wavelength. Packets undergo o/e and e/o conversions at every intermediate node. Contention at nodes is resolved by using small elastic store-and-forward buffers. In a variant of this approach [22], store-and-forward buffering is eliminated by routing data on a single wavelength at high bit-serial data rates.

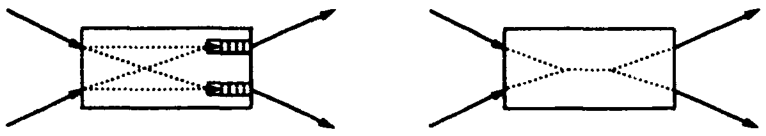
A common feature of all the approaches outlined above is that the wavelength domain is utilized to multiplex multiple bit-serial data coded on different wavelengths onto the same fiber. An alternative to this approach is to utilize the available transmission spectrum for parallel transmission of bits of the same data item using bit per wavelength encoding as used by

Sauer in the Boulder UFO network [57]. The architecture supports the optical flow-through paradigm through static bufferless, deflection routing at nodes. Packet-switching is implemented with minimal optoelectronic conversion at switching nodes, with the payload data maintained in optical format. The various architectural and design issues involved in developing the photonic nodes for an extended packet-switched fiber optic backplane for multiprocessors are dealt with in the thesis by Blumenthal [58].

2.3 Photonic Deflection Networks

The deflection routing protocol is essential to take advantage of the potentials offered by optics for efficient interconnection. A major consideration in the design of optical interconnects is that optical fibers provide cheap bandwidth and low loss transmission over longer distances as compared to electronics. Optical logic is expensive, and it is not possible to provide static optical storage now. If storage is needed at intermediate nodes, currently each node must perform optical to electrical and electrical to optical conversion on all messages passing through it, thereby slowing the network. Hence, it becomes mandatory to use a routing mechanism, such as deflection routing, that avoids static storage, and maintains the advantages provided by optical transmission. Nevertheless, limited optoelectronic conversion is needed to implement the routing control processing using fast pipelined electronic processors, thereby avoiding optical logic.

Deflection routing [7] provides an alternative routing strategy to the traditional store-and-forward routing used in packet-switched networks. In networks that operate under a store-and-forward routing paradigm, packets that are not able to exit through the desired output port at a node are stored in



(a) A store-and-forward node (b) A deflection routing node

Figure 2.1. A schematic representation of store-and-forward and deflection routing nodes

an intermediate buffer until the required outgoing link is available (Fig. (2.1.a)). The main idea of the deflection routing strategy is to minimize the necessary processing and storage overhead at each node by routing a packet out through an alternate output port rather than holding it until the desired output port is available (Fig. (2.1.b)). Such alternate routing requires the network topology to be such that there are multiple paths between nodes in the system. Packets will continually move along in a *flow through manner until they reach* their destination. Thus, the whole network will behave like a dynamic buffer. The HEP multiprocessor system [8] used this strategy. Recently, a lot of interest has surfaced on this routing strategy, because deflection routing does not require asynchronous storage of through going packets and hence is well suited for lightwave flow through architectures.

The fundamental requirement of deflection routing is a topology in which there is at least one path from every node to every other node. In a network with such a topology deflected packets can eventually reach their respective destinations. A variety of topologies such as Shuffle Exchange network(SXNet)[59], ShuffleNet [19], Toroidal Net [60], and Hypercubes [61] satisfy these requirements. There is a penalty associated with each deflection. This penalty is an increase in source to destination transit time through increased residence time in the network. It arises due to traversal of more internode links than necessary, as opposed to the residence time overhead at intermediate nodes in a store-and-forward network due to storage of packets contending for output links. The penalty for each deflection is dependent on the network topology. The effect of the penalty on the latency of packets tends to become severe as the internode distance is increased.

Deflection routing can be used in fiber optic packet-switched networks in which data is packetized for either bit-serial or bit-parallel transmission. Optical fibers can support very high bit-serial data rates of the order of 10Gb/s. The straightforward way to implement deflection routing in fiber optic networks is to code the packet on a single wavelength for high speed bit-serial transmission. Photonics possess an additional degree of freedom, provided by the wavelength domain, over electronics. The common approach that utilizes the wavelength domain, is to multiplex multiple bit-serial data coded on different wavelengths onto the same fiber for transmission. While such multiplexing is useful for point to point transmission, it cannot be used in conjunction with deflection routing due to the complexity of the resulting network nodes. Multiplexed packets need to be demultiplexed at the inputs to separate the different bit-serial streams, contention between packets on the same wavelength must be resolved and the outputs on each wavelength must be multiplexed on to the output fiber on each channel.

Another possibility is the multihop approach. It utilizes wavelength to effect the routing in a logical network that is superposed on a physical broadcast star topology. As mentioned earlier, the original multihop technique [10, 19], routed packets from the source node to the destination node via intermediate nodes in a sequence of *hops* on the fiber, each hop using a different fiber channel. The different channels were created in the fiber through assignment of a fixed number of wavelengths to the transmitters and receivers at each node, such that a logical connection is established between pairs of nodes that transmit and receive on the same wavelength. If deflection routing is to be implemented in such networks, it becomes necessary to shift a packet coming in on one wavelength into the wavelength required for transmission without optoelectronic conversion. Presently, the multihop approach is made possible by converting packets into electrical format and reconverting them into optical format at another wavelength, while routing data on a single wavelength at high bit-serial data rates.

A common feature of all the approaches outlined above is that the wavelength domain is utilized to multiplex multiple bit-serial data coded on different wavelengths onto the same fiber. An alternative to this approach is to utilize the available transmission spectrum for parallel transmission of bits of the same data item using bit per wavelength encoding as used in the Boulder UFO network [57]. The architecture supports optical flow-through paradigm through static bufferless, deflection routing at nodes.

2.4 The Boulder Ultrafast Fiber Optic Network

The approach to exploiting the high bandwidth of fibers in the Boulder Ultrafast Fiber Optic Network is to transmit a word in parallel over a single fiber using different wavelength channels , i.e. each bit in a word is transmitted at a different optical wavelength. The control is separated from payload by coding payload bits at one group of wavelengths (e.g. about $1.55\,\mu$m) and the control bits at another well-separated group of wavelengths (e.g. $1.3\,\mu$m). An example of the format in the optical wavelength domain is shown in Fig. (2.2)

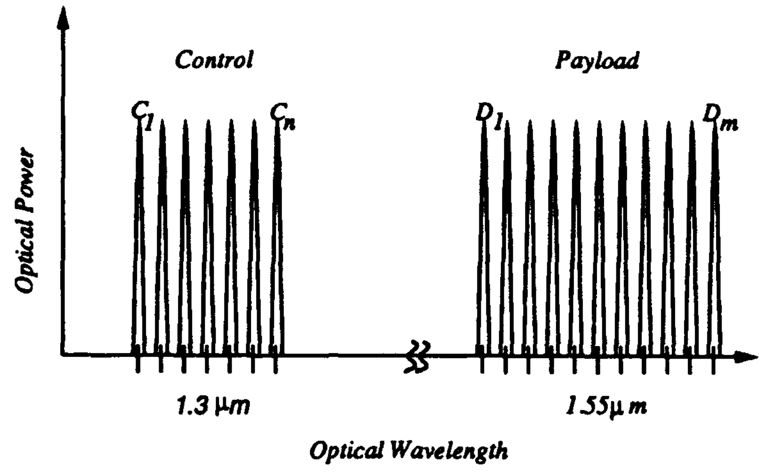


Figure 2.2: An example of Bit per Wavelength Encoding

for n control bits and m payload bits.

Asynchronous storage is avoided in order to utilize the high bandwidth of photonic switches, so the payload data is delayed using a delay line while the routing processor computes the switch settings. The control processing is implemented in high speed electronic logic. Thus, limited optoelectronic conversion is used to extract the control data for processing. The control processor is pipelined so that one control setting can be produced every network cycle [62]. Static buffering is avoided by using the deflection routing (hot-potato) protocol to handle output port contention. If deflection routing is not used, payload data must also be converted into electronics at nodes for storage and reconverted to optics for transmission. Such electro-optic conversion at every node is expensive and will necessarily slow the network.

The network nodes are connected to form a ShuffleNet [19], which yields a packet flight time that grows slowly with size of the system. An individual switch node is designed to permute a set of optical network inputs plus a local host output to a set of optical network outputs plus a local host

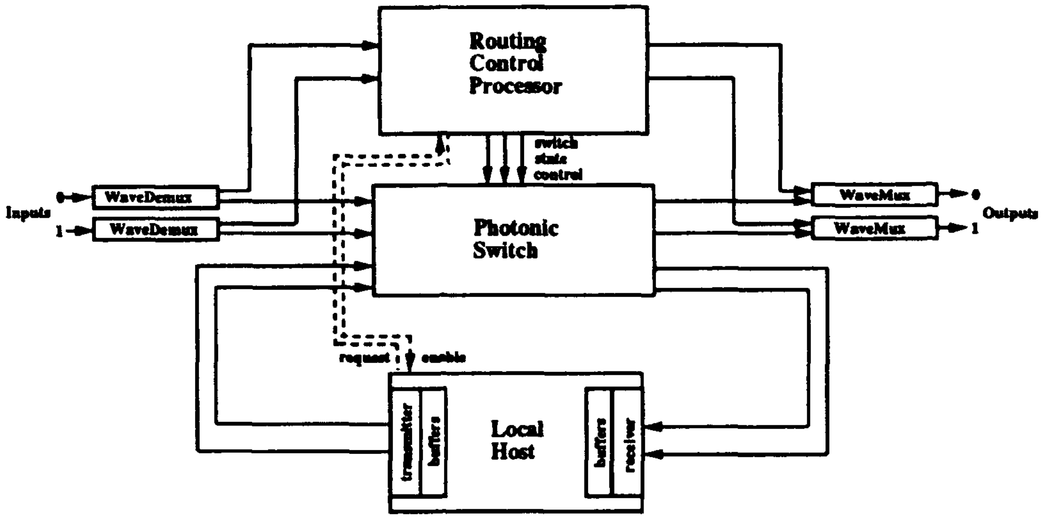


Figure 2.3. Block diagram of a switching node showing its functionality (from thesis by Blumenthal)

input as illustrated in Fig. (2.3). Each node consists of a photonic switch, a routing control processor (RCP), and a local host [63]. Solid lines indicate fiber optic connections and dashed lines indicate electronic connections. Switching elements must route wideband optical data at relatively high reconfiguration speeds. Typical switch bandwidths should be compatible with optical amplifier bandwidths (e.g. 25 - 100 nm). Reconfiguration rates should be on the order of 1 ns. Switch crosstalk must be kept to a minimum since data is amplified at each node output and may travel through several nodes before reaching its destination.

Control of the switch is handled by a parallel pipelined processor which operates on optical control information extracted from each input. The individual control bits are separated at the control processor using a demultiplexer capable of resolving the individual wavelengths. Since photonic switches can have very high bandwidths, the complete parallel payload portion can be switched to one of the two output ports with one device. Optical data signal

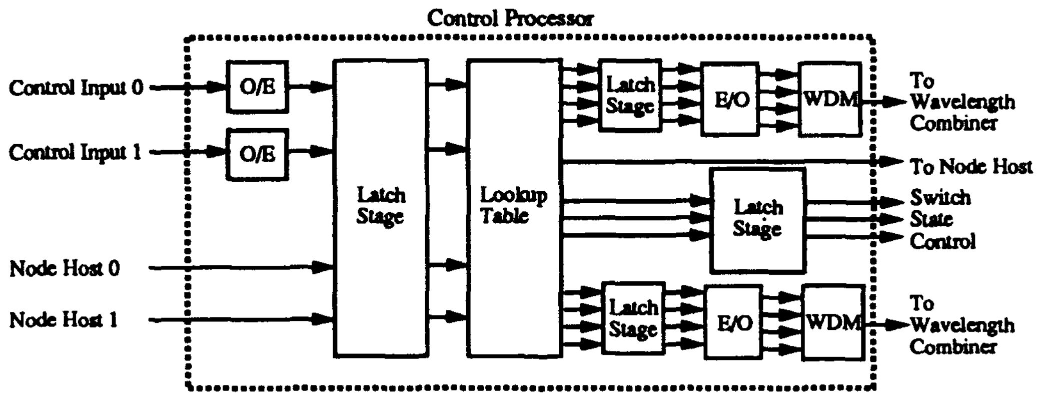


Figure 2.4. Block diagram of a parallel pipelined switch control processor (from thesis by Blumenthal)

levels are restored by optical amplifiers at the node outputs. A single optical amplifier is used for each output. The optical control information is regenerated by the control processor and combined with the payload portion at the node outputs. The maximum processing rate or throughput is limited by the slowest processing element in the pipeline. Pipelining of the routing requests allows packets to be served as they enter the node without elastic buffering in the data path. The control processor has simultaneous access to requests from both network inputs and both node host output queues.

An example lookup table type RCP is shown in Fig. (2.4). The processor arbitrates access to the network from the node host by disabling one or both of the host output queues when full packets are present at the node inputs. The processor also arbitrates contention for the two outputs to the network. The control processor is responsible for detecting and generating both electronic and optical control signals. We use electronically controlled photonic switches since they are readily available. Electronic signals are used to set the switch state and regenerated optical control signals are passively combined with the output data to form new packets which are transmitted to the next node. Thus only the minimal routing information is electro-optically regenerated while the full word-wide optical payload is routed through the switch while remaining in optical format. A complete treatise on the multiwavelength node design can be found in [58].

2.5 Topological Studies on Deflection Networks

A number of research groups have analyzed deflection networks under certain specific conditions. The basic approach used has been to use techniques developed by Greenberg and Goodman [64]. They developed two models, one for node based analysis and the other for packet based analysis to compute the steady state throughput and average packet delay for an uniformly loaded network and presented results for a 64 node MSNet. The MSNet is a 2-dimensional toroidal network proposed by Maxemchuck [12] and is receiving wide attention. Greenberg et. al.'s work as well as that of others based on his technique, make certain approximations that reduce the complexity of the underlying Markovian model while producing reasonable results for unform loads.

Krishna and Hajek [11] have used similar techniques to study the performance of shuffle-like switching networks with deflection routing. Their study includes the SXNet as well as ShuffleNet, each of them being a special case of a generalized class which they call the Shuffle Ring network. The performance parameter considered in this paper is the *evacuation time*, which is the time it takes for the network to evacuate when operating synchronously with no packets injected other than those of the initial load. They derive a set of evolution equations which determine the behavior of a typical packet in the system. These evolution equations were then used to derive bounds on evacuation time for the network. They have also considered variations of shuffle networks for reducing the negative effects of deflection. One of these, the cross-back switch, uses shuffles augmented by inverse shuffles, while the other uses a shuffle augmented by the identity interconnection. They concluded that improvement in performance may barely compensate the increased complexity of the switches for both variations.

Choudhury and Li [65] have developed a variant of the one node model of Greenberg et. al. also for uniform loading conditions under random resolution rule for MSNet with and without buffers. They have presented analytical as well as simulation results for a 64 node network. Their conclusion was that the MSNet performs well under uniform heavy load and that the introduction of a few buffers improves the throughput and delay in the network to a great extent.

In [66] Brassil and Cruz have considered a simple abstraction of nonuniform traffic in a MSNet using a model similar to all of the above and have developed the model for random as well as other routing rules. Assuming that host and link states are independent random variables, they have derived analytically the probability distribution of the residence time of a packet in the network. They created a hot spot in a 4×4 MSNet by generating requests uniformly at a rate of $1/15$ packets per slot at all but a specific node that was the destination node for all requests. An iterative solution to the model produced approximate delay distribution and link utilizations. They also illustrate through an example how deflection routing forces packets to circumvent congested regions in the network. To do this, they created nonuniform traffic by producing a hot spot at an intermediate node. Exactly 2 sources were active in an 8×8 MSNet, with each source generating one packet per network cycle for a distinct destination, such that they have a common node along the shortest path to their respective destinations. Using their model, they computed the link utilization of the links showing the spreading of packets over a large number of links. The hot intermediate node is entirely occupied routing transit packets, thereby being blocked from injecting packets.

Bannister et. al. [21] have developed a model, based on a weaker independence assumption than in [66], to arrive at the mean packet delay for arbitrary network topologies under arbitrary traffic pattern. Their approach also develops a set of nonlinear equations that need to be solved. The procedure, which they call internal-link-flow algorithm, yields results that agree very closely with simulations. This approach is computationally simpler than that used by Brassil and Cruz, but cannot model delay distribution.

Deflection routing in Hypercube networks has been studied by Greenberg and Hajek [13] and Szymanski[14]. While all work cited so far has dropped packets that cannot enter the network as soon as they are created, Greenberg et. al. consider queuing of excess arrivals until they can be admitted to the network. They analyzed both the transient and steady state behavior of the network and simulated the network to compare the analytical predictions with simulated behavior. They found data from simulations were close to predictions. The general conclusion that was reached was that deflection routing works well in hypercube networks. Under steady state operation, the mean queuing delay was small compared to transit delay and the delay due to deflection is minimal. As in the case of modified networks proposed in [11] the efficiency in performance comes with increased complexity of the switching nodes.

Apart from the studies cited above, a simulation based study on comparison of deflection and store-and-forward techniques in the MSNet and SXNet under uniform load conditions has been presented by Maxemchuck [67]. In another simulation study, Robertazzi and Lazar [68] have studied deflection strategies in a MSNet and have demonstrated the need for input buffers in a practical implementation.

Researchers at Columbia University are involved in the development of multihop lightwave networks [10]. Their focus is on the ShuffleNet. In [22] they present an analytical comparison of their proposed network under hot-potato routing and the equivalent store-and-forward network. Their analysis of hot-potato routing includes a *time-out* period following every injection into the network during which no other packets may be placed onto the network by the same host. A similar *time-out* occurs at removal of a packet from the network during which no other packet may be removed by the same host. Both these *time-outs* are a feature of the way their lightwave network is implemented. Their analytical approach leads to a set of non-linear equations for network parameters that are solved iteratively. In [20], they present a similar comparison ignoring the effect of time-outs and under the assumption that both the hot-potato network and the equivalent store-and-forward network have the same link speeds.

Ayanoglu and Cabellaro [69] have used signal flow graph technique for path enumeration in multihop networks with deflection routing. Their analysis included the solution of deflection probability as a function of arrival rate of packets.

The Approximate Mean Value Analysis technique [70, 71] allows modeling of entire multiprocessor systems as closed queuing networks. An analytical model of buffered interconnection networks, based on this technique, is presented in [72]. Wagner [73] has extended the technique to bufferless deflection routed networks. The model was applied to a ShuffleNet under uniform traffic and verified through simulations. An important limitation of the technique has been noted to be the computational complexity, which makes analysis of large systems (over 100 nodes) prohibitively expensive.

Recently, Chlamtac and Fumagalli [74] have proposed an all optical switch architecture for use in communication networks and analyzed its performance in a MSNet. They present a number of delay line solutions to the problem of reducing contentions. Their basic approach is to provide a shared, dynamic output buffer at network nodes. The technique presented in the paper is an extension of separate dynamic output buffers suggested by them in [75, 76]. Though their technique is also a form of space-time switching, it is basically different from the space-time switching technique proposed in this work. Their approach to utilizing the time domain is not suitable for fast packet-switches needed for multiprocessor interconnects. A detailed comparison between the two works will be more appropriate after our analysis and findings are presented, and hence is given at the end of Chapter 4.

2.6 Contention Resolution in Deflection Networks

A variety of contention resolution strategies can be adopted to decide the winning packet. The simplest strategy to analyze, though not the simplest from the implementation point of view, is the random resolution rule. The advantage of random resolution is that it is not necessary to maintain the state of the packets for contention resolution. Also, this rule lends itself easily to mathematical analysis. The main drawback of this rule is that there is a possibility of unlucky packets staying in the network for a very long time. The tail of the delay distribution curve under random contention resolution rule is much longer than with other deterministic rules outlined below.

Deterministic contention resolution rules include distance from the destination [59], the residence time in the network [77] and the number of deflections suffered [8]. Unfortunately, there is no consensus on terms used to denote these strategies. The same term, for example *age*, is used by different groups to quantify different parameters related to a packet's residence in the network.

The HEP [78] supercomputer had a counter field in the packet header that contained the age of the packet measured by the number of deflections the packet had suffered. In case of contention, priority was given to the older packet. The results presented in [77] for the Horizon [27] considered pure aging, i.e. residence time in the network, as a time stamp mechanism for assigning priority to packets. The number of deflections was called *battle scars*.

Robertazzi and Lazar [68] have studied various deflection strategies in a MSNet. Apart from random contention resolution, which they call *standard deflection*, they consider a number of deterministic rules. In what they term as *age deflection* strategy, when two or more packets prefer the same output link, the oldest packet, i.e. the one that has been in the network longest, is given preference. Under their *conflict deflection* strategy, deflection decisions are based on the number of deflections a packet has undergone. *Conflict deflection* can be implemented with a deflection counter field with fewer bits than the hop counter field of age deflections. They have also considered several *distance based deflection* routing strategies. In their *static deflection* scheme, a new packet arriving at a node is assigned a priority equal to the shortest distance between its source and destination. This priority is not changed as the packet transits the network. In their *dynamic deflection* scheme, the distance of a packet from its current node to its destination is used in making deflection decisions. Under this scheme, they considered giving preference to the packet further from its destination as well as to the packet closer to its destination.

In [68] too, an arriving packet is cleared from the system if it cannot be accepted immediately into the network. The results presented for an 8×8 MSNet, obtained through simulation, exhibit that both throughput and delay distribution show a similar trend for all contention resolution schemes considered. As compared to standard deflection, age and contention priority rules show a marked reduction in the tail of the distribution. The static deflection strategy has been found to be inferior in shaping the tail of the delay distribution compared to age deflection or even the standard deflection protocol. The dynamic distance resolution also showed a behavior similar to the behavior under static deflection strategy. Another scheme they considered was the *predictive deflection* scheme. It is a hybrid strategy, where each packet is assigned a priority at its entry to the network. This priority equals its source-destination distance. Every time a packet is deflected its priority is incremented by four. During contention, preference was given to the packet with the largest priority. The performance under this contention resolution rule was found to be similar to that under *age* and *conflict* rules.

As mentioned earlier, Krishna and Hajek [11] have studied evacuation times in shuffle like networks under deflection routing. They have considered contention resolution rules such as giving priority to packets closest to their destination, and assigning priority based on the number of deflections a packet has undergone. The second rule was considered to analyze the proposed Boulder UFO network [79], wherein the number of deflections is called the *age* of the packet following the terminology used in the HEP [8]. They found that the evolution equation method works well in predicting delay and throughput for the network in steady state as well. The study concluded that allowing a wider *age* field results in a higher average delay, but reduced the spread of the delay distribution in a ShuffleNet under uniform load.

Szymanski [14] has investigated *priority to closest to destination* rule as well as *random contention resolution* rule in a deflection routed hypercube. He found that, while the former performs better than the latter, the improvements are not significant at light to moderate loads. At heavy loads the first scheme resulted in a slightly lower average delay.

Brassil and Cruz [80] have analyzed the MSNet under uniform traffic under different contention-resolution policies and with multiple packet buffers at each output link. The contention resolution policies that they consider are what they term t *slotcount* rule, the *random* rule and the *straight* rule. The *slotcount* rule gives preference to the oldest contending packet. While *random* rule assigns random priority for contention resolution, the *straight* rule gives priority to the packet closest to its destination. In [66] they present a model to compute the delay distribution and individual link utilizations for deflection routed networks under non-uniform traffic and all the contention resolution rules. Bannister et. al.'s approach [21] can model the *destination proximity* rule, but cannot model an *age-based priority* scheme.

Zhang and Acampora [81] present an analysis of the ShuffleNet under two different priority schemes. In the *distance-age priority* scheme, priority is given to the contending packet close to its destination and if the contending packets are at the same distance from their destination, priority is given to the packet that has suffered the greatest number of misroutes. In the *age-distance priority* scheme, the order of giving preference to distance and age is reversed. Results of both cases were compared against those for random contention resolution. They also considered networks that dropped packets that cannot enter the network as soon as they are generated as well as networks that buffered such packets. The results presented were based on the numerical solution of their model and were not validated by simulation.

CHAPTER 3

THE NETWORK AND THE SIMULATOR

3.1 Network Description

In this chapter we describe our deflection routing network and the organization and features of the simulator that we have built to study it. In this section we present the topology and organization of the networks considered and the performance metrics that are relevant to the characterization of their behavior.

3.1.1 Network Configuration. The networks we consider have switch nodes with 2 inputs from and 2 outputs to other network nodes. A host is attached to each switch node by means of two bidirectional ports. Potentially, a host can inject and/or receive two packets into/from the switch node every network cycle. Since the nodes are effectively 2×2 switches with no buffering capability for storing through going packets, priority is given to route through going packets. A block diagram of the internal organization of a node is shown in Fig. (3.1). A newly generated packet can enter the network, only if either one of the incoming links is free or if one of the incoming packets is for the host. Until then the packet is buffered at the the input to the network. It is assumed that enough resources are available to send a packet that has reached its destination node immediately into the host. Since a packet that has entered the network is continuously moved until it reaches its destination, the latency of packets has two distinct components: one is the wait time in the input buffer to

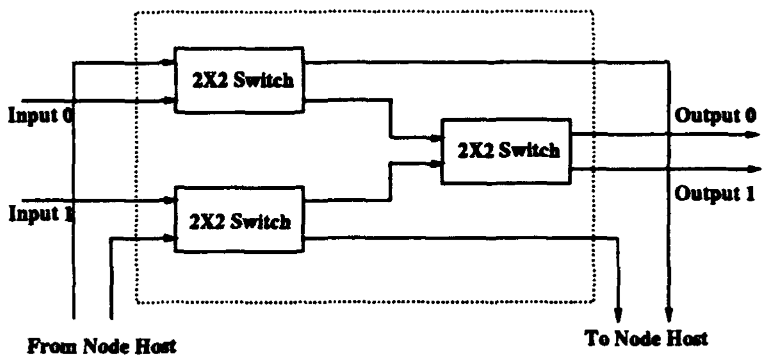


Figure 3.1: Block Diagram of the Internal Organization of a Network Node

the node, and the other is the flight time through the network. For simplicity, in the analysis presented in this chapter, we shall assume that a packet is switched from the input of a node to the input of one of the nodes attached to its outgoing links in one network cycle. We shall call a network cycle a *tick* and the distance traveled by a packet in one network cycle a *tick length.* In reality the nodes as well as the links will be pipelined so that internode transit times will be an integral multiple of network cycle time, and a number of packets can be in transit between two nodes. If all node-link pairs are of constant length, the latency of packets in flight through the network will be proportional to the number of *hops* made, i.e. the node-link pairs traversed, by the packet. The average flight latency of packets, measured in *hops*, will be independent of the internode link length. While flight latencies can be measured either in *ticks* or *hops*, the total latency should be expressed in *ticks.* Internode distances will be expressed in *tick length.*

The ShuffleNet and the MSNet are two widely different topologies that are suitable for implementing deflection routing in networks. While the

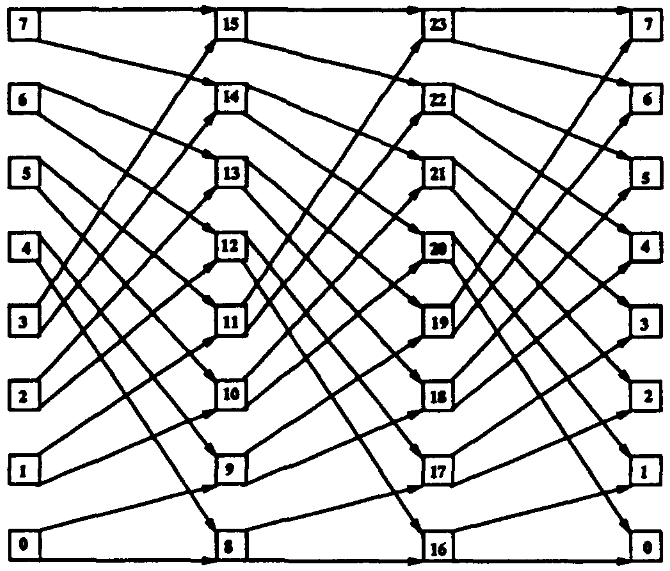


Figure 3.2: A 24 node ShuffleNet interconnect

ShuffleNet provides a network in which the diameter or the average internode distance grows logarithmically with size, the MSNet topology leads to a network whose diameter grows as square root of its size. Though the ShuffleNet provides a much lower average internode distance, it has a deflection penalty that grows logarithmically with the size of the network with the MSNet incurring a constant deflection penalty irrespective of the network size.

A typical ShuffleNet consists of k columns of p^k nodes with nodes in each column connected to those in the next by a p-way perfect shuffle connection. This topology provides near minimum distance between any source destination pair for a network with $p \times p$ switches. The system that we consider here consists of $N = k2^k$ switches of in-degree and out-degree two into the network. Figure (3.2) shows a 3 column 24 node ShuffleNet. Each deflection increases the number of intermediate nodes visited by the packet by k, i.e. the packet is essentially sent around the network once. Thus, the penalty for deflection increases as the log_2 of the size of the network.

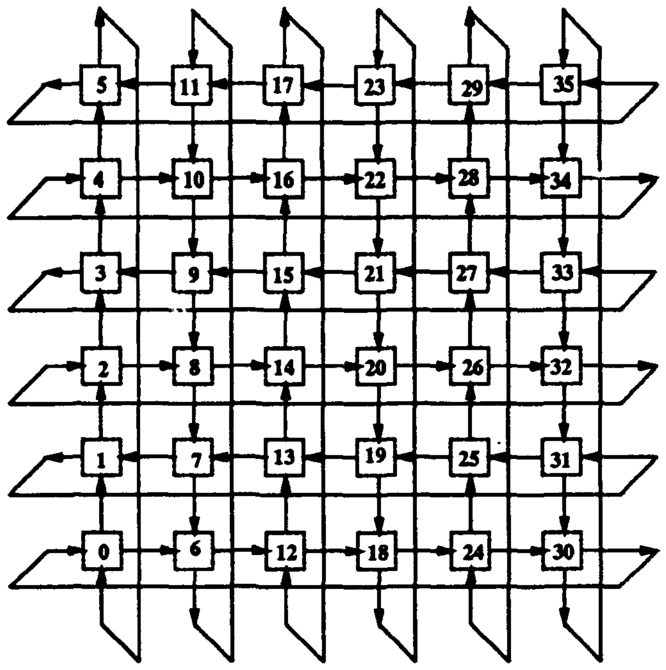


Figure 3.3: A 36 node MSNet interconnect

The MSNet is an unidirectional 2-dimensional toroidal network with the direction of network links alternating between adjacent rows and columns. Figure (3.3) shows a 36 node MSNet with 6 rows and 6 columns. In order to maintain the diameter as low as possible, the network should be as close to a square as possible. Then its diameter is approximately square root of the number of nodes. The penalty for deflection is always 4 *hops*, irrespective of the size of the network. Each deflection causes a packet to visit 4 more nodes.

Both the ShuffleNet and the MSNet are multiple minimum distance path networks. Every node in these networks have some destinations nodes that are at an equal distance from either output port. Thus, the router at each node will encounter packets that are either *care* packets that need one of the two ports to reach their destination in minimum time or *don't care* packets which can be sent out through either port.

3.1.2 Performance Metrics. The node hosts exchange packets of information through the interconnection network. The most important performance measures are the time required for such an exchange and the amount of information transferred. Obviously, the most desirable characteristics are low information exchange time and large information transfer. Some of the metrics defined below are general characteristics applicable to any network, while others are specific to deflection networks. Also, not all of the metrics are independent of others. The important metrics are:

Latency: Latency is the time between the placement of a packet by the source in the network input buffer and the time it is delivered to its destination. As mentioned earlier, in deflection routing networks latency is the sum of two distinct components, the wait buffer latency and the flight latency.

Throughput: The throughput or user bandwidth is the rate at which packets are injected into and received from the network. The network capacity is the sum of the throughputs of all the users. The interaction of raw hardware rates, network size, topology, routing protocol, and the nature of the load on the system will determine the bandwidth available to the user.

Link Utilization: Link utilization is the ratio of the average number of packet slots occupied in the network to the total number of packet slots available. According to Little's Law [82], under steady state conditions the link utilization is the product of the average flight latency and the average throughput. Switch output-port contention at the nodes increases the latency through longer flight time caused by deflections. The longer the packets tend to stay in the network, the smaller will be the throughput of the network.

Network Efficiency: Network efficiency is the ratio of the useful link bandwidth to the total link bandwidth. The useful link bandwidth is the bandwidth needed for satisfying the communication requirements of the load on the system, while the total bandwidth is the sum of the useful link bandwidth and the bandwidth wasted on deflected packets.

Collision Probability: Collision probability is the probability that packets that are routed by a node simultaneously will prefer the same output port. It depends on the volume and uniformity of requests generated by the hosts well as the topology of the network.

Probability of Deflection: Probability of deflection is the conditional probability that, given that a packet *cares* about the output port assignment, it gets deflected. As will be seen later, the probability of deflection is useful in characterizing the latency behavior of packets. The number of deflections suffered by a packet, and hence its flight latency are dependent on the probability of deflection.

3.2 SPRINT: The Simulator

SPRINT is a Simulator for Packet Routing INTerconnects oriented towards word size transfers. It is a generic simulator for self routing multihop, packet-switched, direct networks developed for this study. The development of the simulator as well as the simulations were carried out on DEC and Sun workstations. We have chosen to develop the simulator in a general purpose language to take advantage of the flexibility such languages offer in the design and formulation of the model of the system to be studied, the type and format of the output reports to be generated, the ability to maximize efficiency, and the kinds of simulation experiments performed with the model. The simulator, written in C++, has a modular organization that allows the level of detail in any component module to be increased or decreased as desired without affecting other modules. The structure provides hooks to configure the network under any topology that satisfies the basic requirements. The network size, internode distance and packet size have been parameterized. The simulator can handle a variety of routing protocols such as the hot-potato/deflection and store-and-forward protocols. The host interface design permits the network to be driven by user defined host modules, which can generate probabilistic access patterns created by random number generators as well as access patterns from real traces.

3.2.1 The Data Structure. Figure (3.4) shows the basic data structure used in the simulator. Each entity in the figure is a class. An arrow from a class to another indicates that the latter is an element of the former. A double arrow from a class to another indicates that the former is derived from the latter. A node-host pair and the associated network interface unit (NIU) have been configured as a supernode (SuperNode) with pointers to the network node (NetNode), the host (Host) and the NIU as elements. The desired network (Net) is implemented as a class (ShuffleNet, MSNet, etc.) that contains pointers to all the supernodes, and functions that specify the interconnect topology to hook the supernodes. The connections between nodes are implemented using queues whose heads are connected to the destination of a link, and the tails are connected to the source end of the link. The queues representing internode links are accessed through pointers that are elements of the NetNode class. The output queues of preceding nodes are connected to

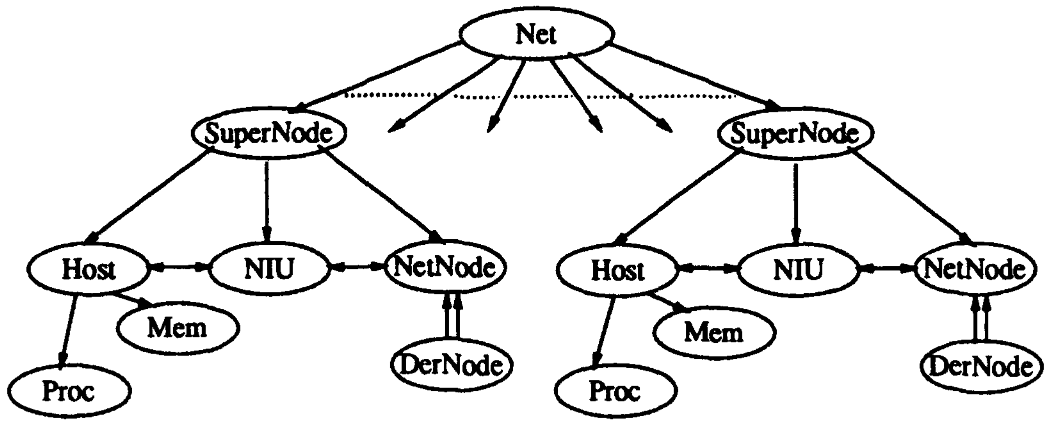


Figure 3.4. A schematic diagram of the basic data structure used in the simulator.

the tails of the corresponding input queues to the following nodes. The output port requirements at every node, to reach all other nodes in the network, are precomputed at network setup time and stored in the NetNode data structure. The NetNode class is a generic class that contains information necessary to carry out routing in a generic network node. Derived classes are used to implement the specific routing protocol such as hot-potato routing. The NetNode class contains a virtual function for routing, which is overwritten by the derived class DerNode.

Messages, in the form of packets, are objects that are moved from their sources to their destinations according to the routing functions implemented in the various modules. The Packet data structure contains general information of interest to the host and the network nodes as well as host specific and network specific information. The NIU isolates a host from the associated network node. The host as well as the network node have bidirectional access to the NIU as indicated by the double sided arrows between the NIU and the Host/NetNode in Fig. (3.4).

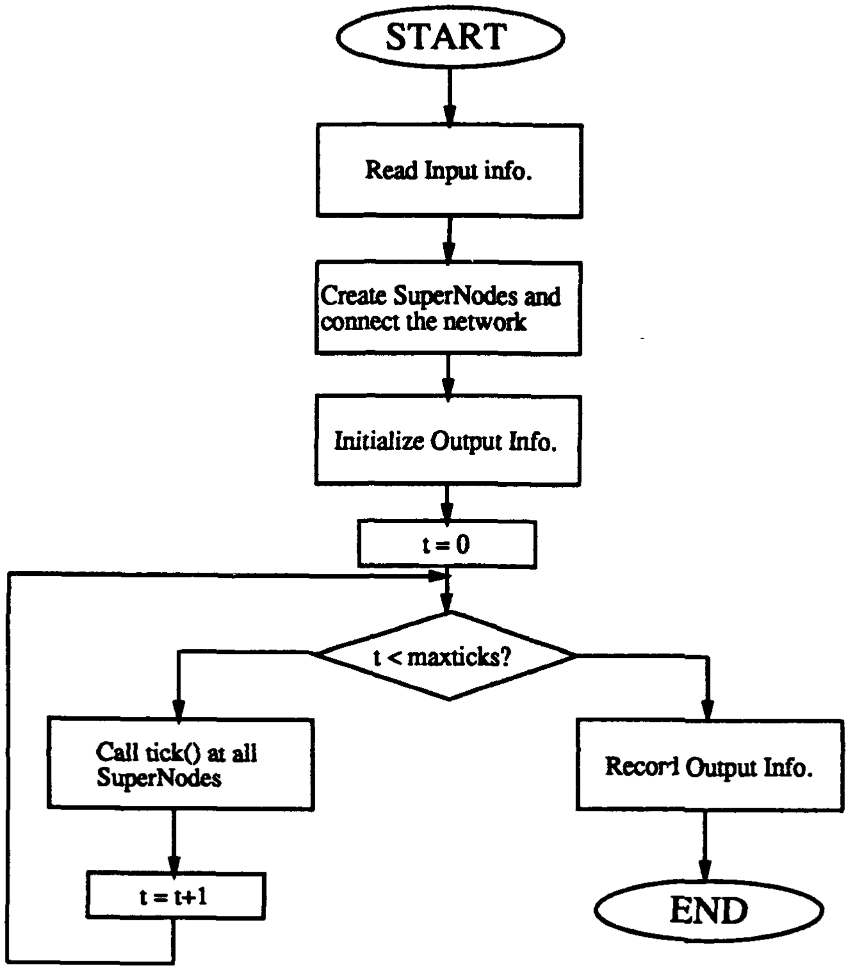


Figure 3.5: The main module of the simulator.

3.2.2 The Functionality and Timing. The simulation model is a discrete event model since the state transitions of our system are deterministic. A network cycle is the unit of time. We use a unit-time advance mechanism in the simulator, since there is a high probability of something happening at every supernode during every network cycle. Figure (3.5) shows the functionality of the main module of the simulator. During every network cycle, the host as well as the network node at every supernode are called upon to perform their functions for that cycle. The functionality of the component modules have been highly abstracted. Packets are moved from one module to the next by inserting their pointers in the appropriate queue with a time stamp to indicate when they should be processed next. When a packet is ready to exit a queue, it is processed as required and placed into the next queue. Output data, specific to the packet, are carried around by the packet structure and are recorded before a packet is destroyed on reaching its final destination. NetNode and Host are provided with appropriate functions to collect statistics during simulations. The largest simulations, using probabilistic work loads, took about 40 minutes in real time on a DEC station 5000/200 to step through 5000 network cycles or ticks. Simulations using traces run much longer. Typically, they run either up to the end of the trace file or until they are timed out. The time out facility was included for debugging during developmental stages. The traces ranged from about 6 million to 30 million ticks.

CHAPTER 4

TWO DEFLECTION NETWORKS UNDER UNIFORM LOAD

4.1 Network Description

The response of a network is a dynamic function of a number of factors including the topology, size, routing protocol and the nature of the load on the system. In this chapter we consider networks under a deflection routing protocol and a random contention resolution strategy. Random contention resolution often allows easy analysis and, as will be seen later, leads to a closed form solution for the ShuffleNet topology under a uniform load. We present simulation results on ShuffleNet that validate our model, as well as the simulation results on the MSNet which does not lend itself to tractable analysis.

4.2 The ShuffleNet Model

A variety of techniques have been used to analyze the performance of the ShuffleNet. This section presents a new approach that uses the specific property of the ShuffleNet, that at any given time if l is the number of internode links a packet has to travel to reach its destination in minimal time from its current node (which may or may not be the initial node), it requires correct routing only at the last $min(l, k)$ of future hops, where k is the number of columns in the network. Though the analysis will be restricted to switch nodes of degree 2, which is our interest, the approach can be easily extended to higher degree nodes.

4.2.1 The Ideal ShuffleNet. Before analyzing the behavior of the network, we shall consider an ideal abstraction of the network in which packets encounter no contention and hence no deflection. Packets inside such a network will always reach their destination in minimum time. If packets are generated at constant rate with uniform address distribution, then the average number of node-link pairs $\langle E_0 \rangle$ traversed by packets will be equal to the statistically expected number of node-link pairs between two randomly selected nodes. $\langle E_0 \rangle$ can be computed by assuming that every node sends a packet to every other node in the network and obtaining the weighted sum of all possible values for the number of node-link pairs to be traversed. Thus,

$$\langle E_0 \rangle = \frac{N}{N-1} \left(\frac{3(k-1)}{2} + \frac{1}{2^k} \right). \tag{4.1}$$

The maximum number of node-link pairs traversed is $2k - 1$. Figure (4.1) shows a plot of the average number and maximum number of hops made by packets in an uniformly loaded ideal ShuffleNet as a function of the number of columns in the network.

As will be seen later, another quantity of interest is the average number of node-link pairs $\langle C_0 \rangle$ at which the packets will care about the output port, i.e. require a particular output port. Packets which can reach their destination in $i \leq k$ hops will care about their output port assignment at every node they visit, while packets that require more than k hops will care about their output port assignment only during the last k hops. Once again, assuming that every node sends a packet to every other node in the network, we can compute $\langle C_0 \rangle$ as the weighted sum of all possible values for the number of care hops. Thus,

$$\langle C_0 \rangle = \frac{((k^2 - 2)2^k + k + 2)}{(N-1)}. \tag{4.2}$$

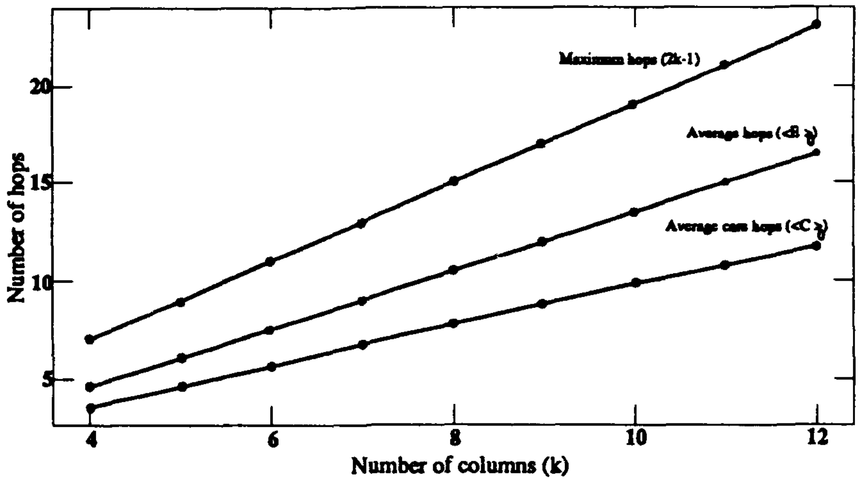


Figure 4.1. Average number of *hops* and *carehops* made by packets inside a ShuffleNet

Figure (4.1) also shows a plot of the average number of *care hops* made by packets in an uniformly loaded ideal ShuffleNet as a function of the number of columns in the network. $\langle E_0 \rangle$ is slightly higher than $3k/2$ and gets closer to $3k/2$ for large networks. Similarly, $\langle C_0 \rangle$ is slightly less than k and tends to k for larger networks. So, as the size of the ShuffleNet grows, about 2/3 of packets routed by a node will be *care* packets.

A N node network has $2N$ internode links. Hence, on an average two packet slots are available for occupancy by packets from each host. Since a packet stays in the network for $\langle E_0 \rangle$ cycles, the average network access bandwidth available to each host γ_0 is given by

$$\gamma_0 = \frac{2}{\langle E_0 \rangle}. \tag{4.3}$$

The system will be able attain a steady state as long as the average injection rate of the hosts is less than this limit. The response of the ideal system will essentially bring out the basic characteristics that primarily depend on network topology and size, by providing the limiting values of throughput and latency that can be achieved.

4.2.2 Real Network. One way to characterize the changing behavior is to consider an ideal network as the baseline and measure the deviations of various metrics under real conditions from that of the ideal system. In a real network, packets will encounter contentions and hence latency will increase due to resulting deflections. In a ShuffleNet, a packet *cares* only at the last $min(l, k)$ nodes on its path to its destination Here, l is the number of node-link pairs in the minimum distance path between the *current* node and the destination of the packet. Figure (4.2) shows the state transition diagram of packets in a ShuffleNet. In this figure, the shaded states represent *care* nodes and those not shaded represent *don't care* nodes encountered by packets on their way to their respective destinations. The states 1 through $2k - 1$ are the possible start states for the packets. A packet which starts at a node at a minimum distance i from its destination, which we call an *i-packet*, will start from state i. The final state of packets is D. The effect of deflections is two-fold: Each deflection will cause a packet to go around the network one more time thereby increasing the latency by the time required for k hops; also, deflected packets utilize network resources that would otherwise be available for new packets waiting to enter the network. As a result, the network access bandwidth available to the hosts will decrease and packets will have to wait longer in the input buffer.

We assume the hosts generate uniform requests at constant rate. Further, contention resolution at nodes is assumed to be random, i.e. packets have no priorities attached to them. Under these assumptions, the probability of deflection of a packet p_d will be independent of the state of the packet and will

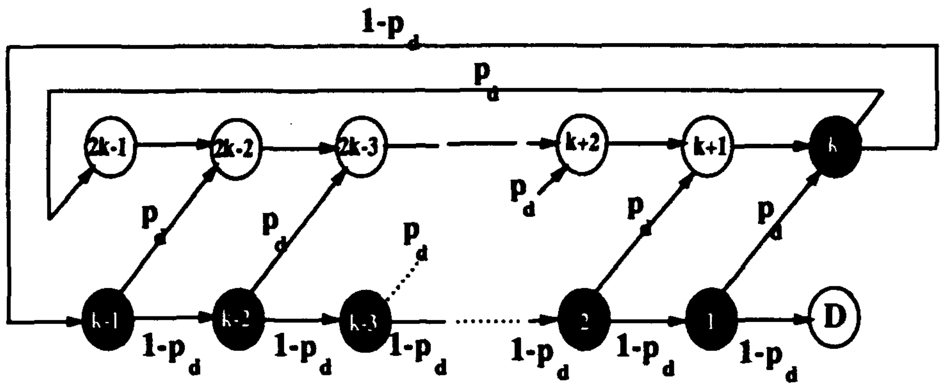


Figure 4.2: The State Transition Diagram of Packets in a ShuffleNet

be the same at all nodes. The flight latency of an *i-packet* is obviously i hops if it is never deflected. If it is deflected within the i hops, the probability of which is $(1-(1-p_d)^{min(i,k)})$, it can reach its destination in $i+k$ hops provided it is not deflected in the last k of the $i+k$ hops. A packet that is deflected twice will reach its destination in $i+2k$ hops, one that is deflected thrice will reach in $i+3k$ hops and so on. In effect a packet can circulate forever, though the probability of doing so goes down with each pass around the network. The probability that a packet will not be deflected during k consecutive hops is $(1-p_d)^k$. Thus, the expected flight latency E_i of an *i-packet* is

$$\begin{aligned} E_i &= i+(1-(1-p_d)^{min(i,k)})[k(1-p_d)^k+2k(1-(1-p_d)^k)(1-p_d)^k \\ &\quad +3k(1-(1-p_d)^k)^2(1-p_d)^k+\cdots], \\ &= i+\frac{k(1-(1-p_d)^{min(i,k)})}{(1-p_d)^k}. \qquad (4.4) \end{aligned}$$

Under uniform loading conditions, the frequency of *i-packets* with i in the range 1 to $k-1$ is $2^i/(N-1)$. The frequency of *i-packets* that have i in the range k to $2k-1$ is $(2^k-2^{i-k})/(N-1)$. The average number of *hops* made by the packets, under the assumption that the destinations are uniformly distributed, is given by

$$\begin{aligned}\langle E\rangle &= \sum_{i=1}^{k-1} \frac{2^i}{N-1}\left[i + \frac{k(1-(1-p_d)^i)}{(1-p_d)^k}\right] \\ &+ \sum_{i=k}^{2k-1} \frac{2^k - 2^{i-k}}{N-1}\left[i + \frac{k(1-(1-p_d)^k)}{(1-p_d)^k}\right], \\ &= \langle E_0\rangle + \left(\frac{N+1}{N-1}\right)\frac{k(1-(1-p_d)^k)}{(1-p_d)^k}. \qquad (4.5)\end{aligned}$$

Thus, the average number of node-link pairs traversed by the packets is the sum of the minimum expected due to the topology and the overhead incurred due to deflections. As can be seen as the probability of deflection approaches zero the contribution of the second term vanishes. As will be seen later, the maximum value of the probability of deflection under uniform loading is 1/4 and will bound the expected latency on the high end.

The average number of *care hops*, $\langle C\rangle$, can be obtained through an analysis similar to the one used earlier for flight latency. But, the computation of $\langle C\rangle$ is not as simple as that of $\langle E\rangle$. There are an infinite number of paths possible between the source and the destination that arise due to the theoretically infinite number of deflections that the packet can suffer. The computation of $\langle E\rangle$ was simplified because a number of paths have the same path length. So, $\langle E\rangle$ could be computed by summing over all groups of paths of the same size, knowing the size of each group. Since, the number of care nodes in each path that belongs to a group of paths of the same size is different, the computation of $\langle C\rangle$ is complicated. Nevertheless, it is possible to apply the specific feature of the ShuffleNet that the path taken by a packet after reaching a node in the destination column following a deflection is independent of the path traversed earlier, and obtain an expression for $\langle C\rangle$. In order to arrive at $\langle C\rangle$, we need to get an expression for C_i, the expected number of care nodes that an *i-packet* will encounter. The approach to computing C_i is to group physical paths into equivalent trajectories in the state transition diagram of Fig. (4.2) and sum over all trajectories with their appropriate weights. Thus, for an *i-packet*, the number of nodes at which the packet will *care* about the output port assigned is given by

$$C_i = \frac{1-(1-p_d)^{\min(i,k)}}{p_d(1-p_d)^k}. \tag{4.6}$$

Using the weights for each C_i, the average number of *care hops* $\langle C\rangle$ is given by

$$\langle C\rangle = \left(\frac{N+1}{N-1}\right)\frac{(1-(1-p_d)^k)}{p_d(1-p_d)^k} - \frac{(2^{k+1}(1-p_d)^k-2)}{(N-1)(1-p_d)^k(1-2p_d)}. \tag{4.7}$$

As p_d tends to zero, $\langle C\rangle$ tends to $\langle C_0\rangle$.

The number of packet slots occupied at any given time is dependent on the injection rate of the hosts and the flight latency of the packets. In the steady state, the average throughput of the nodes will be equal to the average injection rate of the hosts. Hence, the link utilization α, the throughput or offered load γ, and flight latency $\langle E\rangle$ are related as

$$\alpha = \frac{\gamma}{2}\langle E\rangle. \tag{4.8}$$

4.2.3 Probability of Deflection. As the number of packets in flight inside the network is increased, the probability of collision and hence the number of deflections will increase. Under uniform traffic conditions, the probability of collision will be the same at all nodes and at all times. This probability is based on simultaneous arrival of two packets requesting the same output port and is given by

$$p_{coll} = \alpha\frac{\beta}{2}\alpha\frac{\beta}{2} + \alpha\frac{\beta}{2}\alpha\frac{\beta}{2} = \frac{\alpha^2\beta^2}{2}. \tag{4.9}$$

Here, β is the care probability, i.e. the probability that an incoming packet will care about the output port assignment in order to reach its destination in minimum time. If $\langle C \rangle$ is the number of nodes at which an average message cares about the output port assigned by the router, then β is given by

$$\beta = \frac{\langle C \rangle}{\langle E \rangle}. \tag{4.10}$$

Both $\langle C \rangle$ and $\langle E \rangle$ depend on p_d. When p_d is zero, they are equal to their respective ideal values. Hence β becomes $\langle C_0 \rangle / \langle E_0 \rangle$, which tends to 2/3 for large networks. Under a real load, p_d will be finite, but bounded by the maximum possible value of 0.25 as shown below. For a 384 node ShuffleNet, $\langle C_0 \rangle$ is 5.702, while $\langle E_0 \rangle$ is 7.532 giving a value of β=0.757 for p_d=0. At p_d equal to 0.25, $\langle C \rangle$ becomes 17.96, $\langle E \rangle$ rises to 35.39 making $\beta = 0.51$. It can be verified that the value of β falls monotonically as p_d rises from 0 to 0.25.

In this analysis, we are interested in the probability of deflection of a packet. A packet is deflected when it *cares* which output port it requests, a collision occurs and it loses the contention. Under uniform contention resolution the probability of deflection will be given by half the probability that there is a packet on the other link requiring the same output port. Under the assumptions made in this analysis, the probability of a packet appearing at a node during any given packet slot is the link utilization α, and the probability that such a packet will require a specific output port is $\beta/2$. Thus, the probability of deflection of a packet is

$$p_d = \frac{\alpha\beta}{4} = \frac{\gamma\langle C \rangle}{8}. \tag{4.11}$$

According to the first expression for p_d given by Eqn. (4.11), the maximum possible value for p_d is 1/4, and results when the network is fully loaded and all packets care at all nodes. In a ShuffleNet β can never reach 1 under uniform loading conditions due to the presence of *don't care* nodes, and hence p_d will always be less than 1/4. The final expression for p_d shows that p_d is dependent on itself through its dependence on $\langle C \rangle$, thereby providing a relationship between the probability of deflection p_d and offered load γ.

4.2.4 Model Solution Equation (4.5) for flight Latency $\langle E \rangle$ together with Eqn. (4.8) for link utilization α, Eqn. (4.11) for probability of deflection p_d and Eqn. (4.7) will give the flight latency as a function of offered load γ. Because of the interdependence between $\langle E \rangle$, α and p_d, these equations cannot be solved to obtain a closed form solution. However, an approximate solution for $\langle E \rangle$ and hence γ can be obtained easily by approximating β as $\langle C_0 \rangle / \langle E_0 \rangle$. Such an approximation makes β dependent only on the network size and hence causes p_d to be dependent on γ only through link utilization α. Hence, approximate solutions were obtained for constant link utilizations. The accurate solution for $\langle C \rangle$ and $\langle E \rangle$ were obtained through an iterative solution of the equations. Starting with the offered load γ and the ideal flight latency $\langle E_0 \rangle$, an approximate value of link utilization α was obtained from Eqn. (4.8). p_d was then computed using this α and the β used in the approximate solution and used to solve for $\langle C \rangle$ and $\langle E \rangle$. The $\langle C \rangle$ and $\langle E \rangle$ values were then used to compute improved α and β and hence, a better approximation to p_d. The refinement procedure was continued until $\langle E \rangle$ converges. While the algorithm took less than 100 iterations to converge for sizes up to 384 nodes, it took a little over 200 iterations for a 10240 node network. The short iteration steps lead to a computation time far less than simulation times for larger networks. The model predictions are presented along with simulation results in the next section.

4.3 Simulation Results on the ShuffleNet

The behavior of ShuffleNet under deflection routing with random contention resolution was simulated for network sizes in the range $N = 64$ to $N = 2048$. Packet generation during simulations at each source was a uniform random process. The destination address of the generated packets was also assigned to be uniformly random. The offered load was increased in steps from one simulation to the next until the network reached saturation. In all simulation results that will be presented, the values shown or quoted for different network metrics are steady state values unless specifically stated as otherwise. Figure (4.3) shows the latency-throughput characteristics of a ShuffleNet obtained through simulation. In this figure, latencies have been normalized by the average collision free latency for the network and throughputs have been normalized by the maximum possible throughput with no contentions. The latency behavior is qualitatively the same for the two widely different sizes for which the results are shown. As the load on the system increased from an extremely low load, both flight latency as well as total latency increased slowly until moderate link utilizations. As the network approached saturation at higher loads, the total latency increased sharply as shown by the knee in the total latency curves in Fig. (4.3). In both cases, the 64 as well as 2048 node networks, the network turned non-steady at the highest load that was offered to the particular network.

The flight latency tends to level off. The wait buffer component of

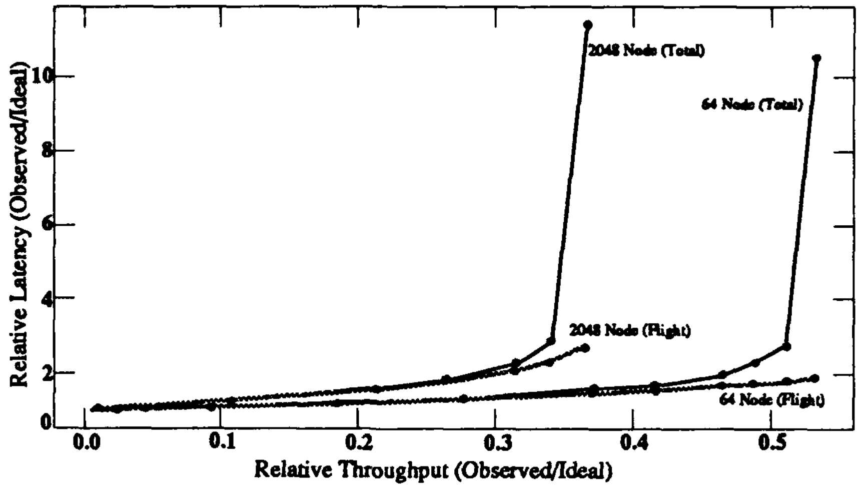


Figure 4.3: Latency - Throughput Characteristics of a ShuffleNet

total latency is low until moderate link utilization, i.e. packets spend very little time in the network input buffer at low to moderate loads. The simulations were also carried out for *internode distances* other than one tick length. Table (4.1) shows the average flight latency and average wait buffer latency in a 64-node ShuffleNet with different internode distances and for two different loads. The offered loads were 0.20 and 0.21 pkts/node/tick, that led to a link utilization of about 0.8 in the first case and 0.9 in the other. It is found that while the flight latency scales with the internode distance, the wait buffer latency remains essentially the same. So, at practical internode distances, the total latency of packets in an unsaturated network under uniform load is about the same as their flight latency.

Figure (4.4) shows a plot of probability of deflection p_d as a function of link utilization α for network of different sizes. Errors computed at 95% confidence interval were small enough to be excluded from the plots.

Figures (4.5) and (4.6) show the flight latency of packets in ticks and the throughput of nodes in pkts/node/tick (packets per node per tick) as a

Table 4.1. Flight and Wait Buffer Latencies in a 64-Node ShuffleNet with different internode distances

Link Utilization approx.	Internode Distance (tick length)	Flight Latency (network ticks)	Wait Latency (network ticks)
0.8	1	8.0	2.4
0.8	10	81.1	2.6
0.8	100	770.3	1.9
0.9	1	8.6	6.9
0.9	10	84.6	6.7
0.9	100	799.5	4.2

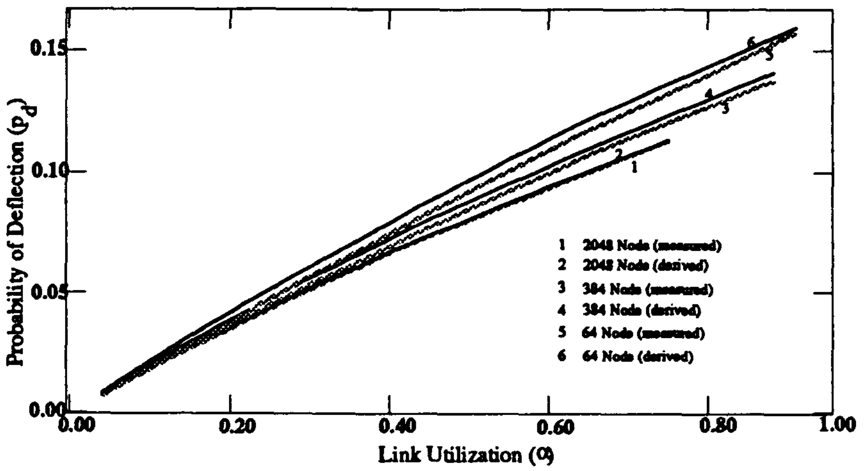


Figure 4.4. Probability of Deflection of a Packet in a ShuffleNet under Uniform Load

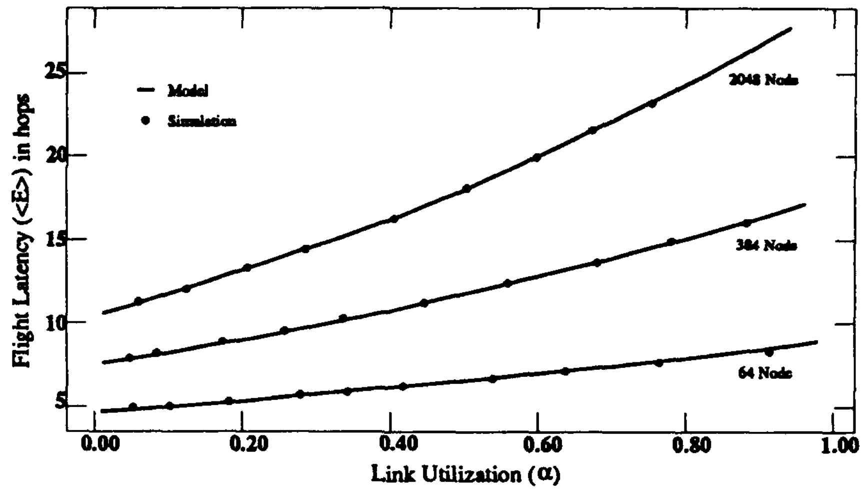


Figure 4.5. Model predictions and Simulation Results of Flight Latency Variations in a ShuffleNet under Uniform Load

function of link utilization. For each network configuration, about 10 data points were generated by simulations. As can be seen, the model presented here provides accurate estimates of average network parameters under uniform loads. The agreement between model predictions and simulation results are good for all three network sizes simulated.

The main advantage of simulations is that they can provide more information than the model yields. For example, while the model provides only average latency of packets, simulations can extract information about latency variations. Such information is particularly important to identify the worst case latencies. Figure (4.7) shows the flight latency histogram for a 384-node ShuffleNet under random and age priority contention resolution rules. Under the random rule, either contending packet is equally likely to win the contention. Under the age priority rule, the older of the contending packets will win the contention. The age of a packet is determined by the number of deflections it has undergone prior to entering the node. In case both contending packets are

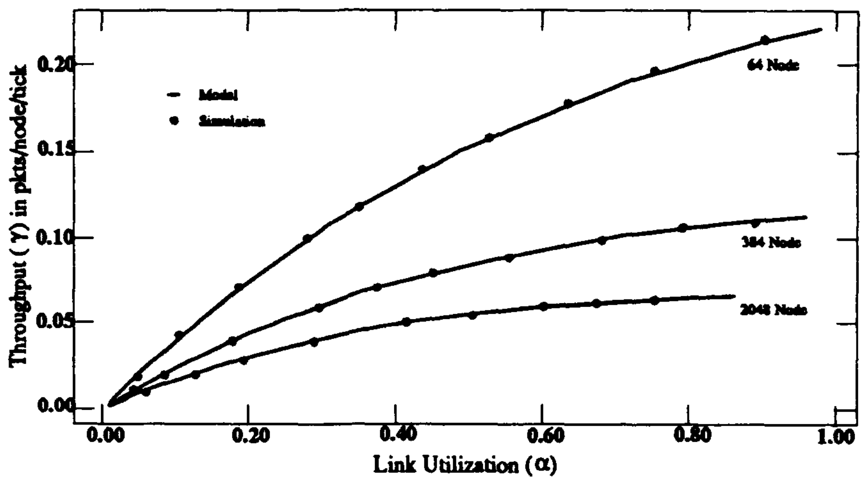


Figure 4.6. Model predictions and Simulation Results of Throughput Variations in a ShuffleNet under Uniform Load

of the same age, one packet is chosen at random to win the contention. The effect of the age priority rule can be seen in the shortening of the tail in the latency distribution shown in Fig. (4.7). As mentioned in Chapter 2, different contention resolution rules have been studied by others, and a similar effect is found in all cases. Though the worst case latency is considerably reduced under age priority scheme, the average behavior is just about the same. The offered load of 0.04 pkts/node/tick in the result presented leads to a link utilization of 0.2. The shortening of the tail is more prominent at higher loads as compared to light loads due to comparatively longer tail under random priority scheme.

4.4 The Manhattan Street Network

The characteristics of MSNet are very different from those of the ShuffleNet with respect to the distribution of *don't care* nodes in the path of a packet. Figure (4.8) shows a 36 node MSNet with nodes labeled by their distance from the node at the left hand bottom of the figure. The minimum distance in this network is 1 while the maximum is 6. In this figure, light

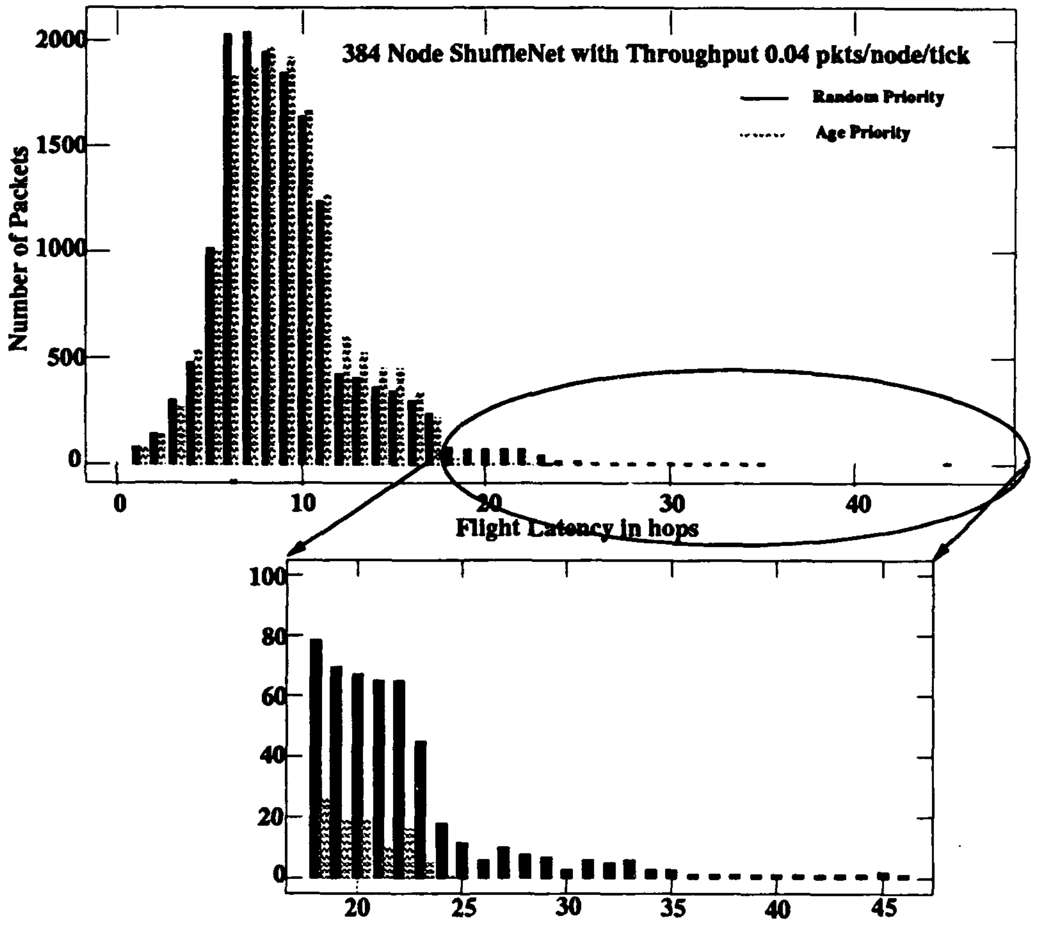


Figure 4.7. Flight Latency Distribution in a ShuffleNet under Random and Age Priorities

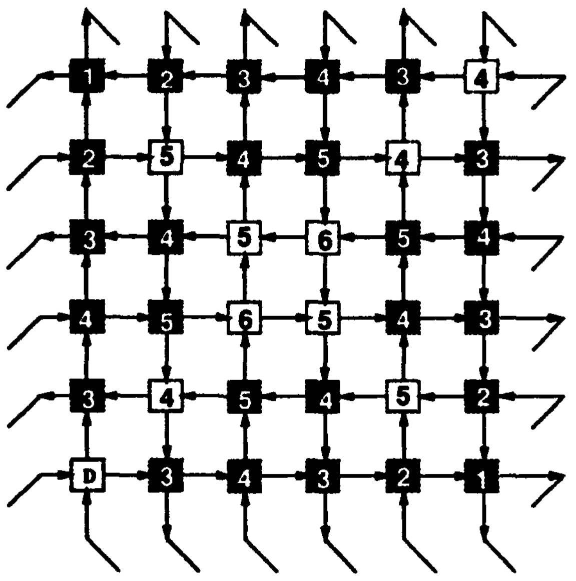


Figure 4.8. The distribution of *care* and *don't care* nodes with respect to the destination node D in a 6 × 6 MSNet

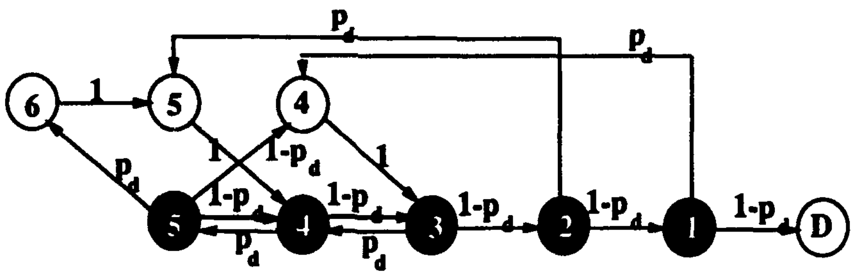


Figure 4.9: The State Transition Diagram of Packets in a 6 × 6 MSNet

nodes are *don't care* nodes with respect to the destination node D, while dark nodes are *care* nodes. While all nodes at a distance 3 or less are *care* nodes with respect to the reference node, some nodes that are 4 or 5 hops away are *care* nodes, while others that are 4 or 5 hops away are *don't care* nodes. Both nodes that are 6 hops away are *don't care* nodes. The state transition diagram for the network is shown in Fig. (4.9). Comparison of this diagram with the state transition diagram shown in Fig. (4.2) illustrates some major differences between the two topologies. In a ShuffleNet, all nodes that are at the same minimum distance from the destination node are in a common state, i.e. either a *care* state or a *don't care* state. But, in an MSNet nodes at the same distance from the destination node do not have a common state. For example, in Fig. (4.9) some nodes that are 5 hops away are *care* nodes, while others at the same distance are *don't care* nodes. In a ShuffleNet, a packet that enters a *care* state continues to pass through only *care* states unless it is deflected before it reaches its destination. In a MSNet, the transitions between *care* and *don't care* states do not exhibit this behavior. As the size of the network increases, both *care* states and *don't care* states subdivide to form more states for packets at certain distances. The distribution of *care* and *don't care* nodes, with respect to a particular destination node D, for a 64 node MSNet is shown in Fig. (4.10). A node with dark boundaries indicates a *don't care* node followed by a *care* and a *don't care* node down the two output links, as opposed to other *don't care* locations which are followed by two *care* nodes. The corresponding state transition diagram is shown in Fig. (4.11). The increase in complexity of the state transition diagrams with size becomes evident, from the large increase in the number of states and possible transitions between them, in Fig. (4.12) for a 144 node MSNet. Note that it becomes necessary to split the care states at distances 5 and 6, because the new states transition into different states. It was not possible to obtain a closed form expression for flight latency even for small networks. Approximations can be made to collapse some states into others. Even if such, approximations are made it is not possible to obtain a closed form expression for latency of packets. The simplest approximate model leads to a set of $O(N^{1/2})$ equations that need to be solved iteratively. Analysis of MSNet using approximate models based on markov processes has been done

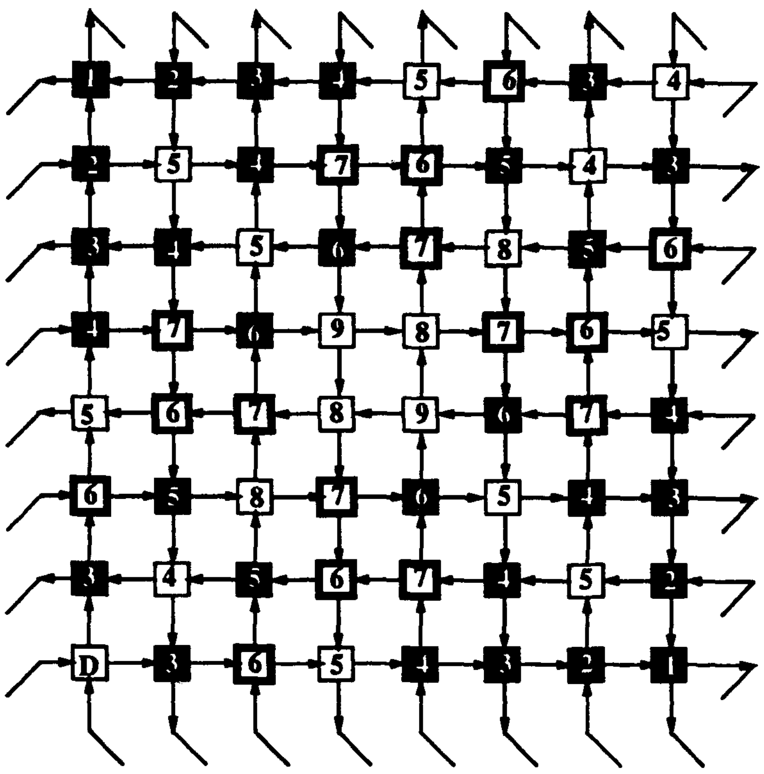


Figure 4.10. The distribution of *care* and *don't care* nodes with respect to the destination node D in an 8 × 8 MSNet

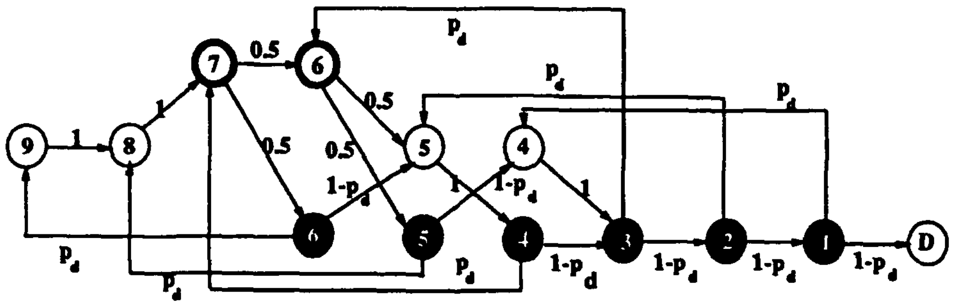


Figure 4.11: The State Transition Diagram of Packets in an 8 × 8 MSNet

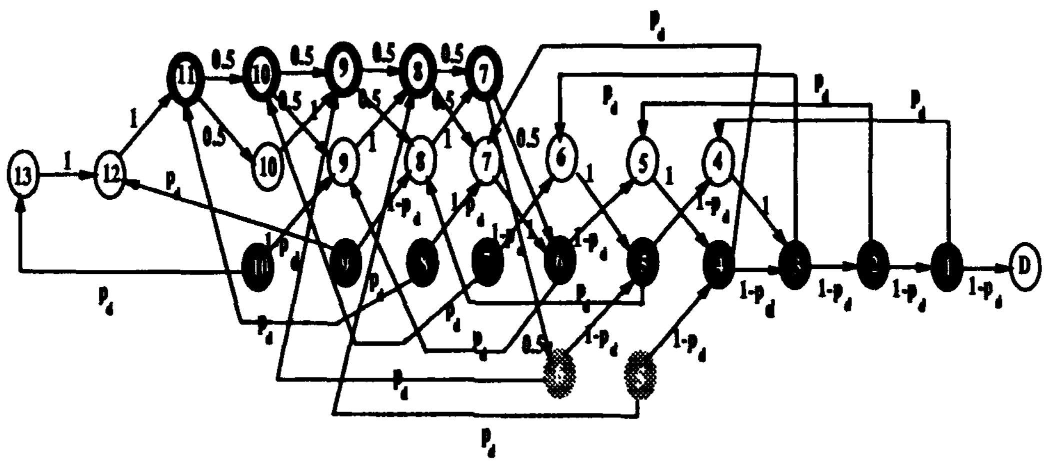


Figure 4.12: The State Transition Diagram of Packets in a 12 × 12 MSNet

by others [64, 65].

4.5 Simulation Results on the MSNet

In this section we present results of simulation of Manhattan Street networks of various sizes. The routing algorithm for deciding the switch settings at nodes in a MSNet is rather complicated. Three routing rules of varying complexity are presented in [83]. The first rule selects the shortest path from any node to the destination node. The scheme uses absolute addresses of the node and its neighbors to compute an output port requirement. The second rule uses a fractional addressing scheme for the nodes and the routing algorithm is simplified by allowing some *care* nodes at the edges of the network to be considered as *don't care* nodes. The third rule uses relative addressing. The third rule is also a simplified rule in which some *don't care* nodes at the edges are considered as *care* nodes. The relative addresses of the node with respect to the destination addresses of the incoming packets is used to determine the switch setting at the node. The third rule has the advantage that it is simple and can be computed in hardware on the fly rather than using a routing table at the nodes. The size of routing table grows with the network and hence, it is desirable to avoid it, if possible. The rule faces the disadvantage that output port contention is increased due to certain simplifying assumptions. As a result, the utilization of one of the output links is slightly more than the other at every node and the path of packets is slightly increased. The trade off is between increase in node processing times due to routing processor complexity and a minimal increase in latency of packets due to simplified approximate routing algorithm.

We have chosen to implement the third rule in the simulator. The rule first calculates the relative address of the current node with respect to the destination node. Since the relative address of a node with respect to itself is $(0,0)$, any other node in a $r \times c$ network will have a relative address (rr, rc) such that $-(r/2) < rr \leq r/2$ and $-(c/2) < rc \leq c/2$. The nodes of the network are assigned to four quadrants Q_1, Q_2, Q_3 and Q_4 according to the relative coordinates as shown in Fig. (4.13). The quadrant of the current node indicates the direction in which to proceed to get to the destination. In Fig. (4.13), the solid arrows are preferred paths and the dotted arrows are alternate paths. A preferred path is a preferred direction towards the destination node. If the current node has a link in that direction, the packet will be able to travel along the shortest path to the destination, if it exits in that direction. The alternate paths in quadrants Q_2 and Q_4 are paths where certain approximations have been made to decide the preference. For example, in quadrant Q_4, links to the left is actually preferred at all nodes except the nodes of relative column 1. In order to treat all columns in that quadrant uniformly, the preferred paths

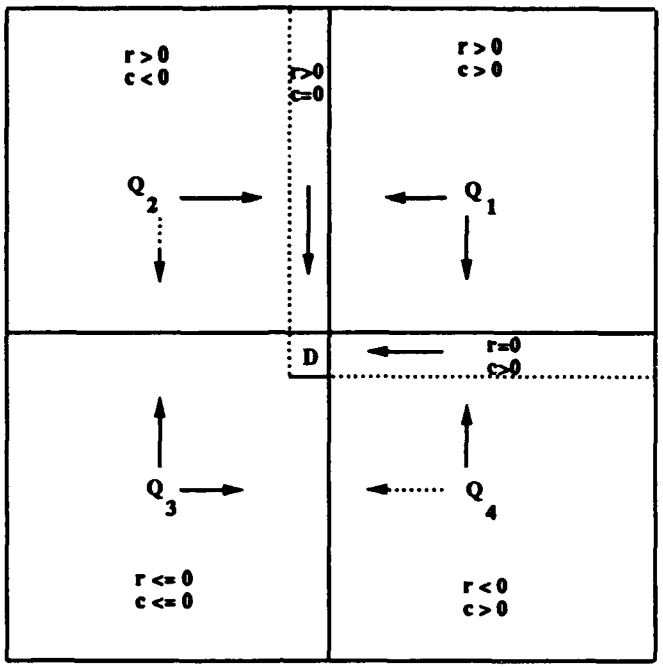


Figure 4.13. Preferred and alternate paths for the routing rule used in the simulations

in all other columns have been designated as alternate paths, which will be taken only if the node does not have a link in the other preferred direction or if the packet loses contention for the other direction. While the preferred and alternate paths have directions as marked, the actual links at nodes may be either up or down along the column and left or right along the row. The routing rule is as follows:

- Select the preferred paths if only one of the output links is in the direction of a preferred path from the current node.
- Select the alternate path if none of the output links is along a preferred direction and if there is an alternate path from the current node.
- Select either path if neither link is in a preferred or alternate direction or if both links are along preferred directions.

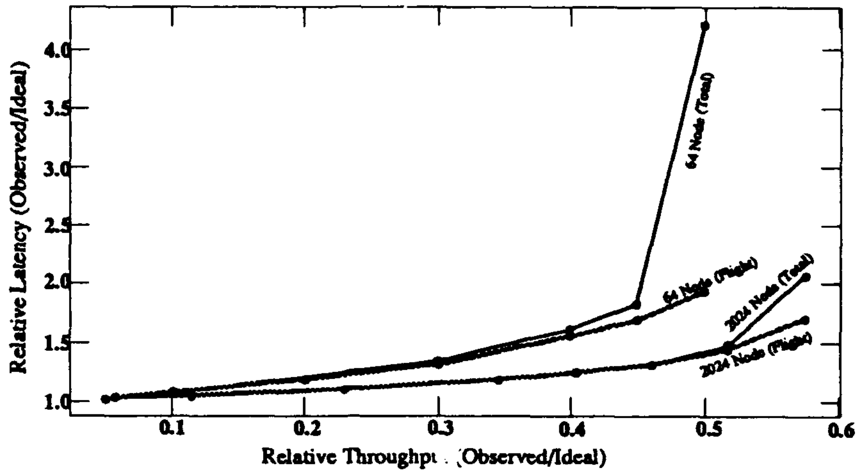


Figure 4.14: Latency - Throughput Characteristics of a MSNet

An 8×8 network (64 nodes), a 20×20 network (400 nodes) and a 46×44 network (2024 nodes) were simulated under deflection routing with random contention resolution. These sizes were chosen to compare their behavior with 64 node, 384 node and 2048 node ShuffleNets. The 46×44 network is an approximate square that is closest to a 2048 node ShuffleNet. Packets generated by sources created an uniform load in the network as was done previously for the ShuffleNet. Figure (4.14) shows the total and flight latency characteristics of the 64 node and 2024 node networks. As in the case of ShuffleNet a knee is seen in the total latency curve. Figure (4.15) shows a plot of probability of deflection p_d as a function of link utilization α for network of different sizes. The latency and throughput variations as a function of link utilization are shown in Figs. (4.16) and (4.17) respectively. Figure (4.18) shows a typical flight latency histogram for a 400 node MSNet under random and age priority contention resolution rules. There is a reduction in the tail of the distribution under age priority rule as against the long tail seen under the random contention resolution rule.

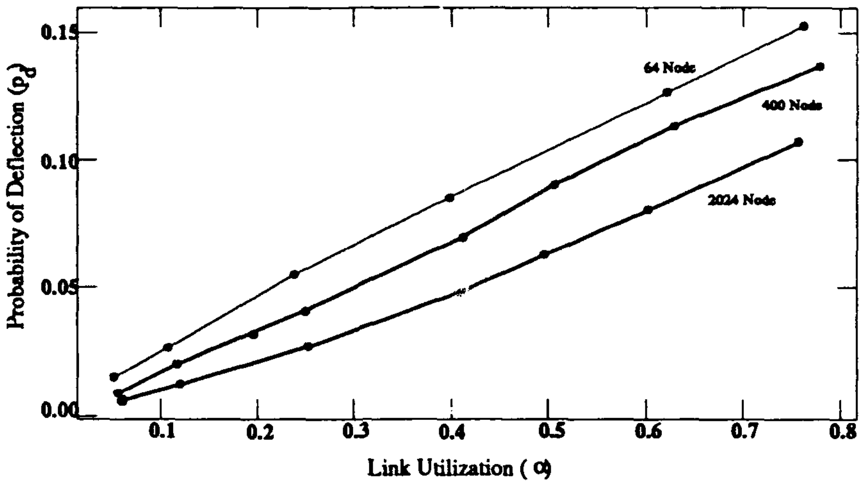


Figure 4.15. Probability of Deflection of a Packet in a MSNet under Uniform Load

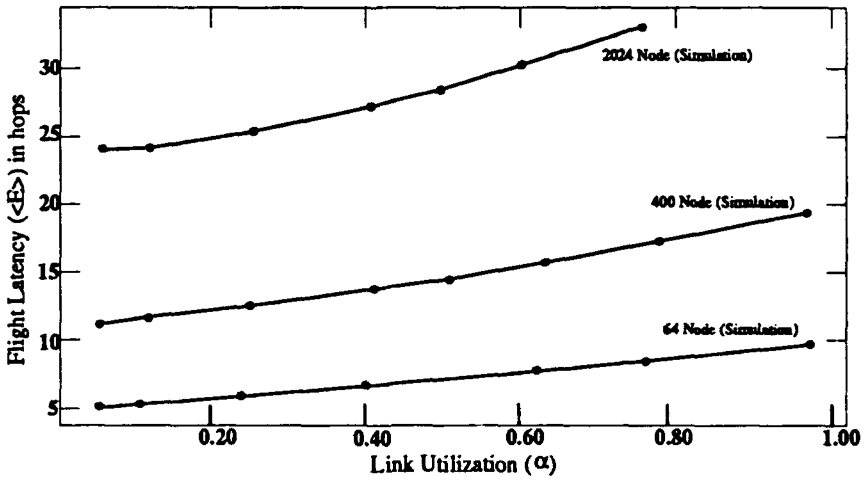


Figure 4.16: Flight Latency Variations in a MSNet under Uniform Load

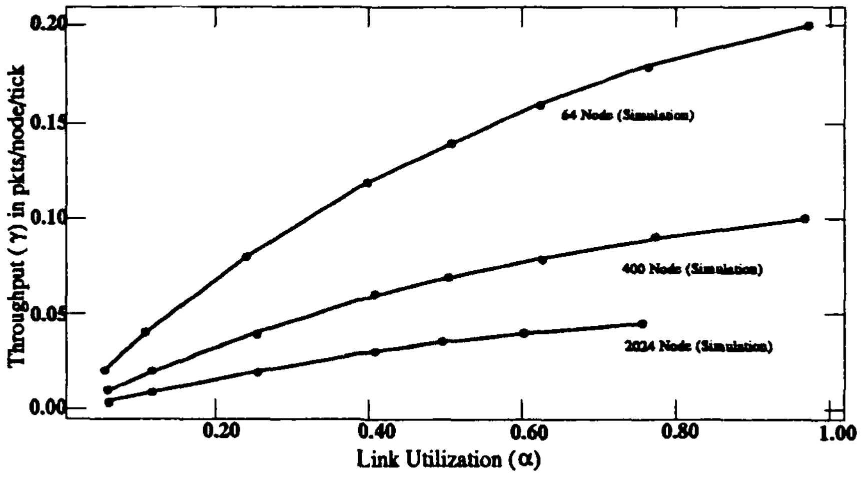


Figure 4.17: Throughput Variations in a MSNet under Uniform Load

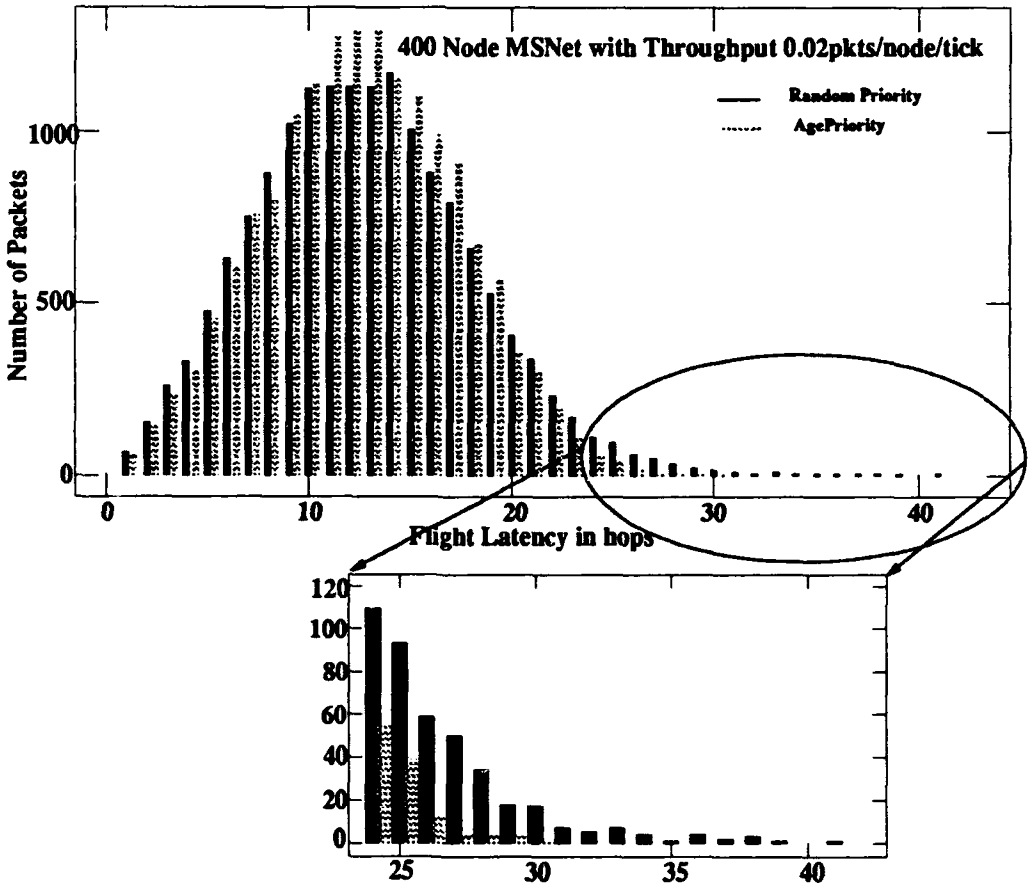


Figure 4.18. Flight Latency Distribution in a MSNet under Random and Age Priorities

4.6 Comparison between the ShuffleNet and the MSNet

The ShuffleNet and the MSNet show the same qualitative behavior under uniform loads. This fact is evident from their latency-throughput characteristics (Figs. (4.3) and (4.14)) as well as the variations in latency (Figs. (4.5) and (4.16)), throughput (Figs. (4.6) and (4.17)) and probability of deflection (Figs. (4.4) and (4.15)) as a function of link utilization. But, a closer examination of the sets of figures cited above shows a vast difference in performance, particularly for large networks. In order to compare the behavior of the two topologies under deflection routing, it is essential to first identify the important network properties that differentiate them. There are two major differences between the two topologies. The diameter or the average internode distance in *hops* in a ShuffleNet of $N = k2^k$ nodes is approximately k. Hence, the diameter of the ShuffleNet increases slowly, as $O(log_2 N)$, with size. In a square MSNet with $N = c \times c$ nodes, the average internode distance is approximately c or $O(N^{1/2})$. The second difference is in the penalty for deflection. The penalty for deflection in a ShuffleNet is k, while that for a MSNet is always 4. While 64 node networks with the two topologies have approximately equal performance due to close values of average internode distance and the same penalty for deflection, the difference in their performance increases rapidly with increase in size. Figure (4.17) shows that a 400 node MSNet can support a maximum offered load around 0.10 pkts/node/tick, which corresponds to a flight latency of about 20 *hops* (Fig. (4.16)) and nearly full link utilization. For the same offered load Fig. (4.6) shows that a comparable ShuffleNet of size 384 gets only about 67% full, with the corresponding flight latency (Fig. (4.5)) being around 13 *hops*. The 384 node ShuffleNet supports a slightly higher load of

0.12 pkts/node/tick with a flight latency of about 17 *hops*. A comparison of the absolute performance of the much bigger 2048 node ShuffleNet and the 2024 node MSNet shows that the maximum latency in a ShuffleNet is comparable to the minimum possible in a MSNet. The reason for this wide difference is that the smaller penalty for deflection in a MSNet is not able to offset the effect of high average internode distance. The effect of the difference in the penalty for deflection can be seen in the relative increase in flight latency in either case with respect to their own minimum value. While the ratio of maximum flight latency to the minimum in a ShuffleNet is little over 2 the same ratio for the MSNet is about 1.5. Another difference to note is in the flight latency distribution of packets shown in Figs. (4.7) and (4.18) for a 384 node ShuffleNet and a 400 node MSNet. Note that even though the ShuffleNet is supporting twice the load being supported by the MSNet, the maximum latency in these figures under random contention resolution is about the same. But the tail of the distribution is shortened far more in the case of the ShuffleNet example than the MSNet example. Such effects also tend to get more prominent for the over 2000 node networks. Thus, the general conclusion is that the ShuffleNet configuration is superior to the MSNet configuration for large networks, at least under uniform loading conditions.

CHAPTER 5

SPACE-TIME SWITCHING IN DEFLECTION NETWORKS

5.1 The Space-Time Switch

This chapter introduces the concept of simultaneous switching in space and time at nodes of a deflection network and presents the analysis of the performance of the networks, studied earlier, under limited space-time switching. A general space-time switch is a multi-space, multi-time channel permutation engine that can reorder packets on the inputs and output them on any channel and in any order. While space-domain switching involves only the switching delay at the switch node, time domain switching involves switching delay as well as a frame delay[84, 85]. The frame delay arises because packets cannot be moved farther than what is possible by the propagation delay. So temporal reordering actually moves a leading packet behind a following packet that needs to move ahead at the output. The minimum delay needed is equal to one slot less than the size of the time window considered for switching. Hence, introduction of a space-time switch instead of a spatial switch at the network nodes will increase the internode flight time of packets by an amount equal to the frame delay, assuming that there is no difference in switching delay between the two alternatives. So apart from the considerations arising from added node complexity and additional hardware cost, the main consideration in choosing the number of spatial channels is the performance tradeoff between increased latency due to frame delay and the reduction in latency due to decreased deflections. We restrict our attention to 2-Space, 2-Time (2S2T) switches so as to reduce the complexity of the node and the node processing time.

Figure (5.1.a) shows a typical input sequence of packets and the corresponding output sequence from a purely spatial 2×2 node is shown in Fig. (5.1.b). A switch with purely spatial switching of the input sequence results in 6 deflections in the output. Figure (5.1.c) shows the corresponding output from a 2S2T switch with fixed frame boundaries, with one such boundary occurring at the extreme right. The reordering of packets in time domain has reduced the number of deflections in the output sequence to 3. If the time window is allowed to slide progressively, the number of deflections is further reduced to 1 as shown by Fig. (5.1.d).

One possible configuration for a 2S2T deflection routing node is shown in Fig. (5.2). The 2×2 deflection routing node is the conventional spatial node with additional capability to compute the control settings for the space-time permuter. The dotted line shows the control line from the basic routing node to the space-time permuter. The space-time permuter is a 2S2T time-slot interchanger shown in Fig. (5.3). The delay lines help to move packets back in time and the 2×2 switches perform spatial rearrangement to collectively implement the necessary permutation. In the configuration of Fig. (5.2), the 2×2 deflection routing node will perform contention resolution between packets arriving simultaneously, enforcing the priority scheme built into it. In case of a deflection which is not repaired by the space-time permuter, the higher priority packet will still remain undeflected. In this configuration, the space-time permuter needs to perform only 3 permutations out of the 24 permutations

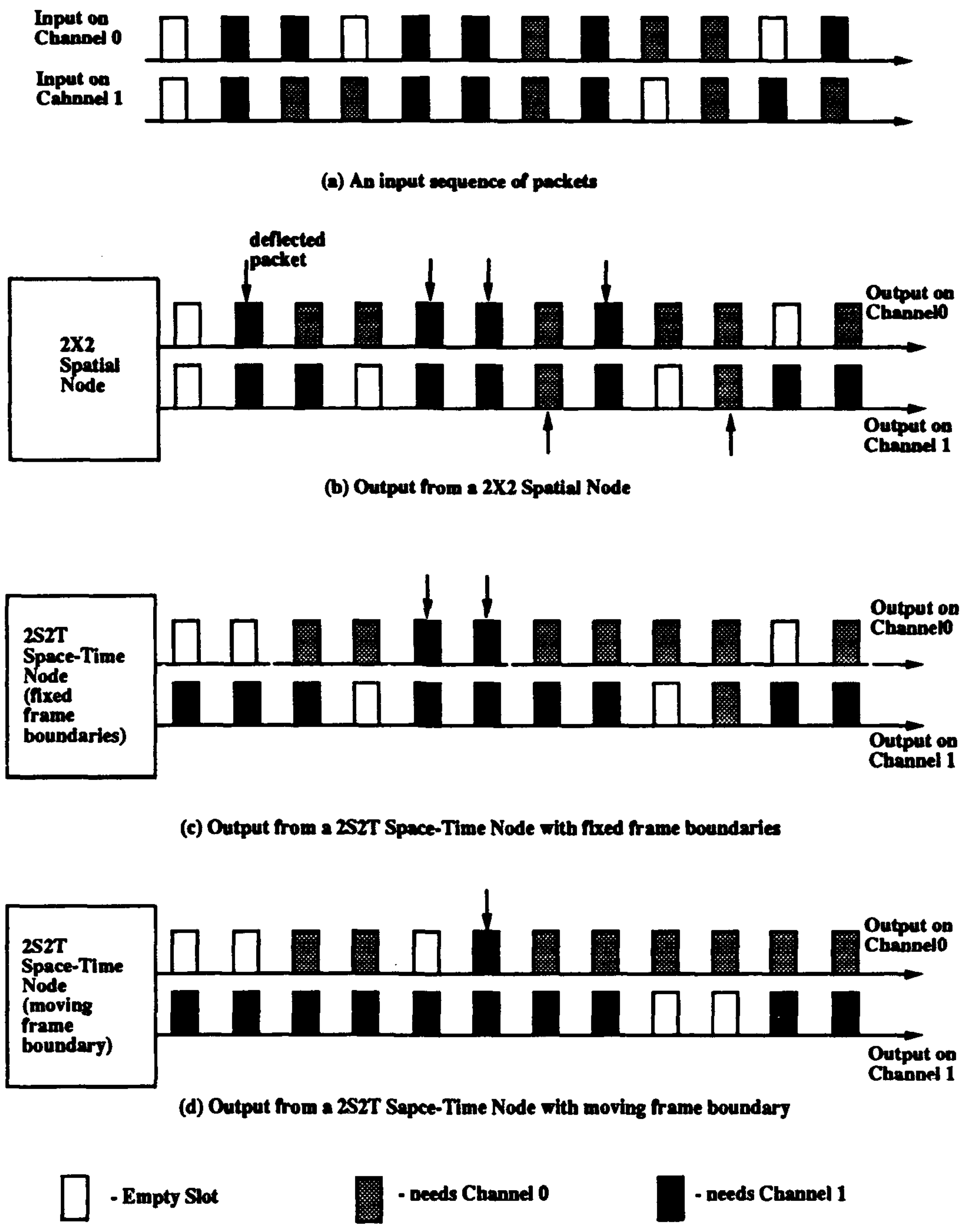

Input on Channel 0
Input on Cahnnel 1
(a) An input sequence of packets
deflected packet
2X2 Spatial Node
Output on Channel0
Output on Channel 1
(b) Output from a 2X2 Spatial Node
2S2T Space-Time Node (fixed frame boundaries)
Output on Channel0
Output on Channel 1
(c) Output from a 2S2T Space-Time Node with fixed frame boundaries
2S2T Space-Time Node (moving frame boundary)
Output on Channel0
Output on Channel 1
(d) Output from a 2S2T Sapce-Time Node with moving frame boundary
- Empty Slot
- needs Channel 0
- needs Channel 1

Figure 5.1. An Example of Reduction in Number of Deflections in a 2S2T Node

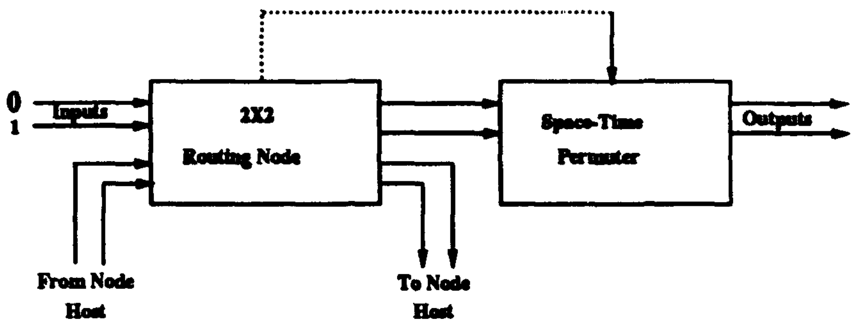


Figure 5.2: A Schematic Diagram of a Space-Time Node

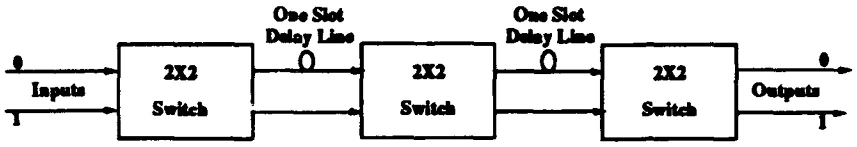


Figure 5.3: A 2S2T Space-Time Permuter

possible in an universal 2S2T time-slot interchanger. The first permutation is an identity permutation, while the other two require swapping of leading packet slot in one of the two spatial channels with the trailing packet slot in the other spatial channel. It is possible to move the switching stage of the 2×2 deflection routing node into the space-time permuter. Such variations may optimize the amount of hardware needed, but is not expected to change the performance of the network node.

5.2 Probability of Deflection in a 2S2T Switch Node

In order to arrive at the probability of deflection of a packet in a deflection network with 2S2T routing nodes, we first consider the contention resolution power of various concurrent packet slot combinations on the two spatial channels. In Table (5.1), the first column contains the various combinations of packets that can arrive at the input during any given time-slot. A "-" denotes an empty slot, a "X" denotes a *don't care* packet,"0" and "1" denote a

Table 5.1. Contention Resolution Power of Concurrent Packet Slot Combinations.

Slot Combination	Probability of Occurrence	can resolve contention for
- -	$(1-\alpha)^2$	any
- X	$\alpha(1-\alpha)(1-\beta)$	any
- 0	$\alpha(1-\alpha)\beta/2$	port 1
- 1	$\alpha(1-\alpha)\beta/2$	port 0
X -	$\alpha(1-\alpha)(1-\beta)$	any
XX	$\alpha^2(1-\beta)^2$	any
X0	$\alpha^2(1-\beta)\beta/2$	port 1
X1	$\alpha^2(1-\beta)\beta/2$	port 0
0 -	$\alpha(1-\alpha)\beta/2$	port 1
0X	$\alpha^2(1-\beta)\beta/2$	port 1
00	$\alpha^2\beta^2/4$	port 1
01	$\alpha^2\beta^2/4$	none
1 -	$\alpha(1-\alpha)\beta/2$	port 0
1X	$\alpha^2(1-\beta)\beta/2$	port 0
10	$\alpha^2\beta^2/4$	none
11	$\alpha^2\beta^2/4$	port 0

packet requiring spatial channels 0 and 1 respectively for optimal routing. The four possibilities on each of the spatial channels create the 16 combinations for possible arrivals during any given time-slot. The corresponding entries in the second column give the probability of occurrence of that combination in terms of the link utilization α and the care probability β. The last column indicates which slot combinations have a capability of resolving a contention for a specific port in the preceding or following time slot. As can be seen, slot combinations "00" and "11" themselves collide and can resolve contention for the other port. All other combinations face no collision, but can resolve either no contention or contention for one particular port or both ports. Figure (5.4) shows a transition diagram for a packet slot combination as it passes through

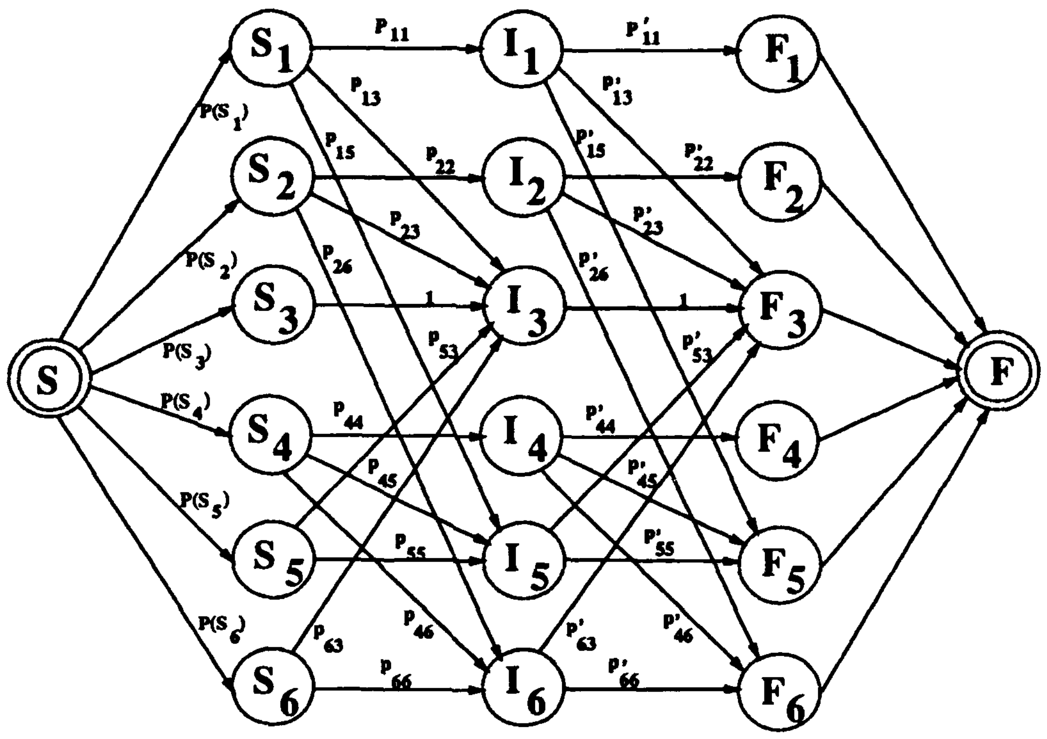


Figure 5.4. State Transition Diagram for a Packet Slot Combination through a 2S2T Node

a 2S2T node. Time domain permutation may replace a packet in a slot combination by another packet from the preceding or following slot combination. We are interested in the net probability of deflection of a packet as it exits a node.

The 16 possible input combinations to the node are reduced to 11 combinations by the first stage, i.e. the spatial 2x2 switch. The grouping of the remaining 11 combinations into the 6 states, S_1 through S_6, of Fig. (5.4) is shown in Table (5.2). $S_1(S_2)$ denotes a packet slot combination in which the incoming packets contend for port 1(0) and can resolve contention for port 0(1). S_3 denotes a combination in which there is no collision and which is not capable of resolving any contention. States S_4, S_5 and S_6 denote combinations in which there is no collision and are capable of resolving any contention, contention for port 0 and contention for port 1 respectively. The arcs leading from the initial state S into the 6 states are marked by the probability of finding an input combination in these states. When an incoming packet combination is passed through the first stage, i.e. the 2×2 routing node in Fig. (5.2), the packet combination will enter one of the 6 start states. On entering the 2S2T space-time permuter they are permuted, if necessary and if possible, with packets in the preceding time slot and reach the intermediate states I_1 through I_6. These states are similar to the start states, except the probabilities of finding a slot combination in these states are different from the corresponding start states. Likewise, states F_1 through F_6 are the corresponding final states after possible permutation with packets in the following time slot with a different probability of being in that state. Table (5.2) shows the different input combinations that go into the states S_1 through S_6, the initial probability of occurrence of these states and the intermediate states reachable from them. Similar transitions are possible from the intermediate sates to the final state. From the node's point of view, at every network cycle the space-time permuter stage encounters a new slot combination in one of the 6 start states and has one other slot combination in flight in one of the 6 intermediate states. It will permute them if necessary and put a slot combination each in an intermediate state and a final state. Table (5.3) shows the state transitions that are performed by the space-time permuter.

In the following, we denote the probability of being in a state Q by P(Q) and the probability of transition from a state S_i into state I_j by p_{ij}. We denote by p'_{ij} the corresponding probability of transition from the intermediate

Table 5.2: Slot Combinations and their Initial and Intermediate States

State	Slot Combinations	Probability of Occurrence	can transition into
S_1	11	$\alpha^2\beta^2/4$	I_1, I_3 or I_5
S_2	00	$\alpha^2\beta^2/4$	I_2, I_3 or I_6
S_3	01	$\alpha^2\beta^2/2$	I_3
S_4	--, -X, X-, XX	$(1-\alpha\beta)^2$	I_4, I_5 or I_6
S_5	-1, X1	$\alpha\beta(1-\alpha\beta)$	I_3 or I_5
S_6	0-, 0X	$\alpha\beta(1-\alpha\beta)$	I_3 or I_6

Table 5.3: State Transitions inside the Space-Time Permuter

Input States	Output States	Input States	Output States	Input States	Output States
S_1, I_1	I_1, F_1	S_2, I_1	I_3, F_3	S_3, I_1	I_3, F_1
S_1, I_2	I_3, F_3	S_2, I_2	I_2, F_2	S_3, I_2	I_3, F_2
S_1, I_3	I_1, F_3	S_2, I_3	I_2, F_3	S_3, I_3	I_3, F_3
S_1, I_4	I_5, F_5	S_2, I_4	I_6, F_6	S_3, I_4	I_3, F_4
S_1, I_5	I_1, F_5	S_2, I_5	I_6, F_3	S_3, I_5	I_3, F_5
S_1, I_6	I_5, F_3	S_2, I_6	I_2, F_6	$S3, I_6$	I_3, F_6
S_4, I_1	I_5, F_5	S_5, I_1	I_5, F_1	S_6, I_1	I_3, F_5
S_4, I_2	I_6, F_6	S_5, I_2	I_3, F_6	S_6, I_2	I_6, F_2
S_4, I_3	I_4, F_3	S_5, I_3	I_5, F_3	S_6, I_3	I_6, F_3
S_4, I_4	I_4, F_4	S_5, I_4	I_5, F_4	S_6, I_4	I_6, F_4
S_4, I_5	I_4, F_5	S_5, I_5	I_5, F_5	S_6, I_5	I_6, F_5
S_4, I_6	I_4, F_6	S_5, I_6	I_5, F_6	S_6, I_6	I_6, F_6

states into the final states.

The probability of deflection of a packet, p_d is the conditional probability that a packet cares about its output port and gets deflected. Since we are considering a uniform work load , we can arrive at this probability by considering a packet that requires one of the two ports. Consider a packet that requires port r for optimal routing. It will face a contention if the incoming slot on the other spatial channel contains a packet that requires port r and hence, the inp., combination is in state S_r. T The probability that the packet will ultimately get deflected is given by

$$\begin{aligned} p_d &= \frac{1}{2}P(F_1) + \frac{1}{2}P(F_2), \\ &= \frac{P(S_1)p_{11}p'_{11}}{4} + \frac{P(S_2)p_{22}p'_{22}}{4}, \\ &= \frac{P(S_r)p_{rr}p'_{rr}}{2}. \end{aligned} \tag{5.1}$$

Since the network is operating under uniform loading conditions, the probabilities p_{11} and p_{22} are equal, so are p'_{11} and p'_{22}. p_{11} is the probability of a packet in the state S_1 transitioning into state I_1, which will happen if it encounters an intermediate slot combination in states I_1, I_3 or I_5. Similarly, p_{22} is the probability of a packet in the state S_2 transitioning into state I_2, which will happen if it encounters an intermediate slot combination in states I_2, I_3 or I_6. Hence, the probabilities p_{rr} and p'_{rr} are

$$\begin{aligned} p_{rr} &= p_{11} = P(I_1) + P(I_3) + P(I_5), \\ &= p_{22} = P(I_2) + P(I_3) + P(I_6). \\ p'_{rr} &= p'_{11} = P(S_1) + P(S_3) + P(S_5), \\ &= p'_{22} = P(S_2) + P(S_3) + P(S_6). \end{aligned}$$

The probabilities of finding a slot combination in the intermediate states I_1, I_3 and I_5 are

$$\begin{aligned}
P(I_1) &= P(S_1)P(I_1) + P(S_1)P(I_3) + P(S_1)P(I_5); \\
P(I_3) &= P(S_1)P(I_2) + P(S_2)P(I_1) + P(S_3) + P(S_5)P(I_2) + P(S_6)P(I_1); \\
P(I_5) &= P(S_1)P(I_4) + P(S_1)P(I_6) + P(S_4)P(I_1) + P(S_5)(1 - P(I_2)).
\end{aligned}$$

The expression for p_{rr} becomes

$$p_{rr} = \frac{P(S_1) + P(S_3) + P(S_5)}{1 - P(S_1)[P(S_2) + P(S_4) + P(S_6)]}.$$

On substituting the probabilities of finding the input combination in states S_1 through S_6, the probability of deflection p_d becomes

$$\begin{aligned}
p_d &= \frac{1}{2}.\frac{\alpha\beta}{2}.\frac{\alpha\beta(1-\alpha\beta/4)}{1-(\alpha^2\beta^2/4)(1-\alpha\beta/2)^2}.\alpha\beta(1-\alpha\beta/4), \\
&= \frac{\alpha^3\beta^3}{4}.\frac{(1-\alpha\beta/4)^2}{1-(\alpha^2\beta^2/4)(1-\alpha\beta/2)^2}. \qquad (5.2)
\end{aligned}$$

Equation (5.2) shows that at full link utilization ($\alpha = 1$), the probability of deflection in a single minimum distance path network falls from 0.25 in a 2×2 switch to 0.15 in a 2S2T switch. The decrease in probability of deflection is much more in multiple minimum distance path networks. Figures (5.5) and (5.6) show the variation of probability of deflection with link utilization and care probability in a 2×2 node and 2S2T node respectively obtained through the preceding analysis. By choosing a topology with as small a care probability as possible and by operating the network at sufficiently low link utilization, the operating conditions in the network be brought to the very low probability of deflection region in Fig. (5.6).

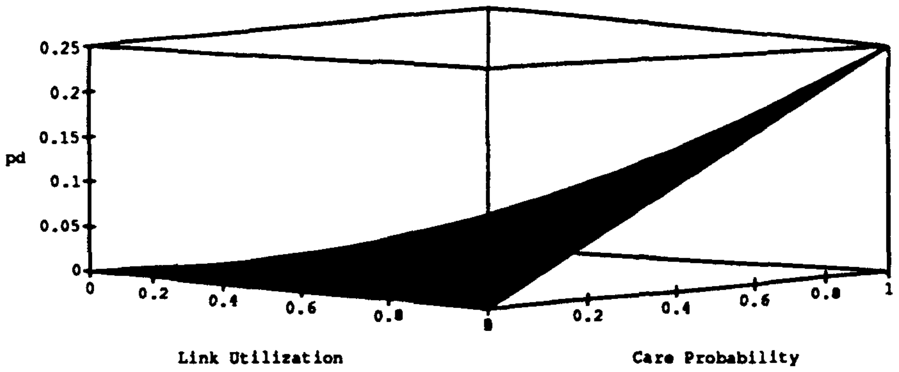


Figure 5.5: Probability of Deflection in a 2X2 Node

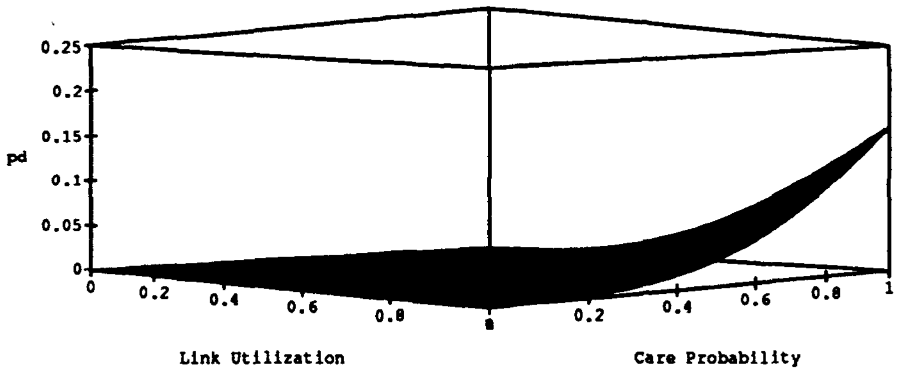
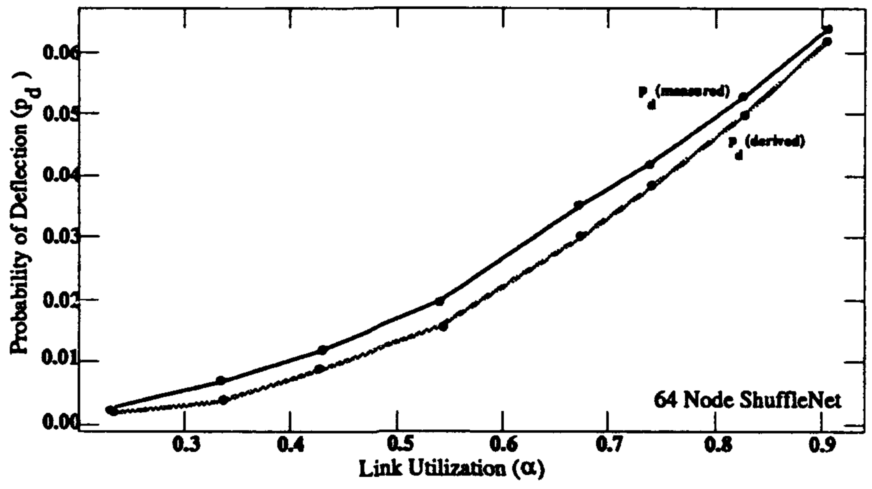


Figure 5.6: Probability of Deflection in a 2S2T Node

5.3 Simulation Results on ShuffleNet with 2S2T Nodes

In this section we present some results on the performance of a ShuffleNet, under deflection routing and space-time switching, obtained through simulation. In a ShuffleNet many source-destination pairs have multiple minimum distance paths between them. In this network the minimum number of node-link pairs between a source and a destination, d, varies from from 1 to $2k - 1$. As shown in Fig. (4.2), a packet will care about the output port assignment only at the last $min(d, k)$ nodes. As a result, a packet that gets deflected will enter a ***don't care*** state until it is again at a distance of k hops from its destination. Thus, the probability of care β is less than 1. For the network sizes that we simulated β varied between 0.6 and 0.8. Figure (5.7) shows the probability of deflection p_d as a function of link utilization α obtained through simulation for a 64-node (k=4) and a 2048-node (k=8) ShuffleNet, under uniform loading conditions. In this figure, $p_d(measured)$ is the probability of deflection measured as the ratio of number of deflections to the number of care visits and $p_d(derived)$ is the probability of deflection calculated using Eqn. (5.2) and the measured values of link utilization α and care probability β. The agreement is close, though consistently the measured values are slightly higher than that expected by application of Eqn. (5.2).

Figures (5.8) and (5.9) show plots of the average flight latency obtained through simulation of a 64-node and a 2048-node ShuffleNet with 2S2T nodes as well as 2×2 nodes. We have chosen to represent the flight component of latency for two reasons. The most important reason is that the introduction of time-domain switching primarily affects the flight latency component of the total latency. As was shown in section 3.3 the wait buffer component of latency



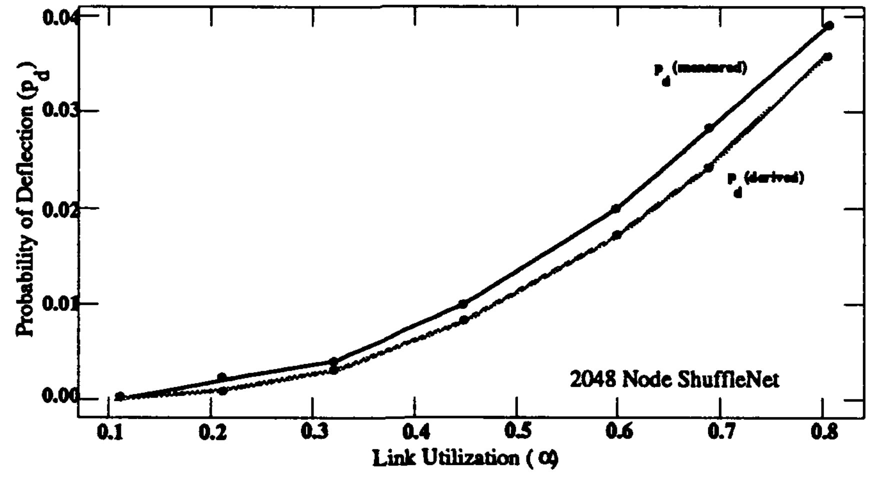


Figure 5.7: Probability of Deflection of a Packet inside a ShuffleNet

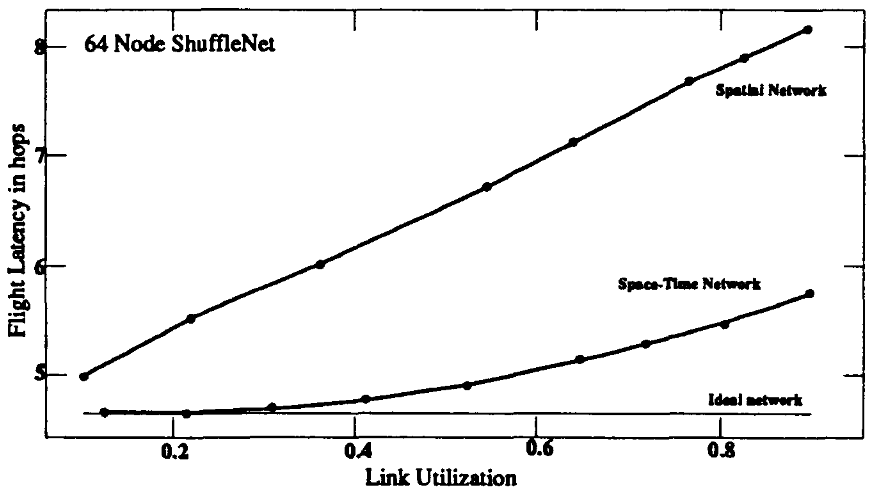


Figure 5.8: Flight Latency vs. Link Utilization for a 64 Node ShuffleNet

is negligible as compared to the flight latency of packets under operating conditions of interest to us. The flight latency in the space-time network remains close to ideal, particularly up to moderate link utilizations. When identical load is presented to the spatial network and the space-time network, the reduction in flight latency of packets in the former will lead to lower link utilization in that case. Hence, the space-time network can support higher throughput as shown in Figs. (5.10) and (5.11), which show the observed throughput as a function of link utilization in ideal, spatial and space-time networks. As with flight latency, the throughput is also close to ideal in a space-time network up to moderate link utilization.

Another important improvement in the network behavior due to time-domain switching is in the flight latency distribution of packets. In a deflection routing network, the tail of the delay distribution tends to be long if random contention resolution is used at the nodes. Any priority scheme, such as providing preference for the colliding packet that has been deflected more, helps to shorten the tail of the distribution as illustrated by Figs. (4.7) and (4.18).

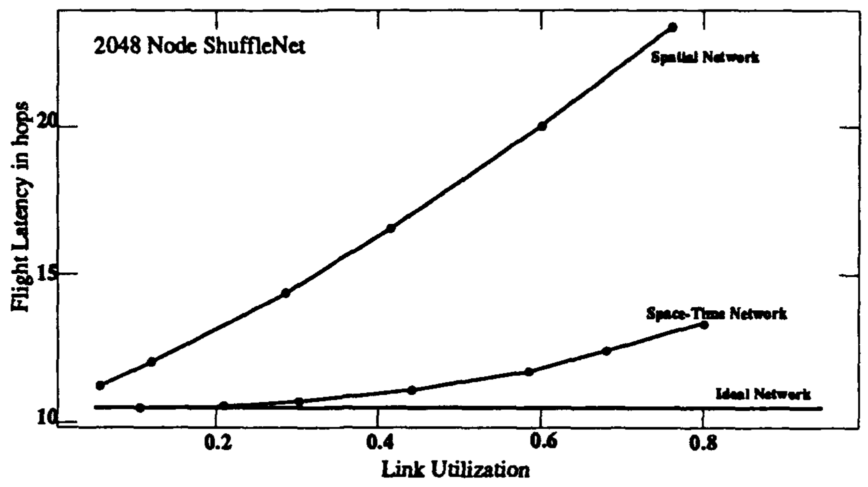


Figure 5.9: Flight Latency vs. Link Utilization for a 2048 Node ShuffleNet

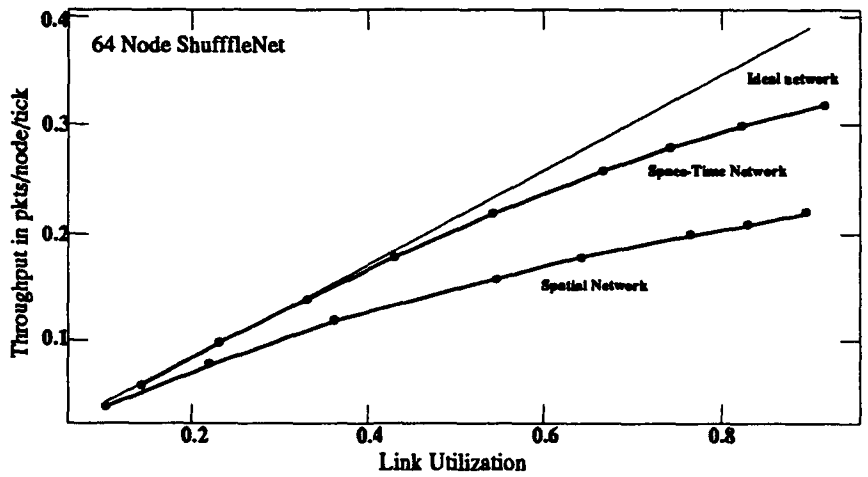


Figure 5.10: Throughput vs. Link Utilization for a 64 Node ShuffleNet

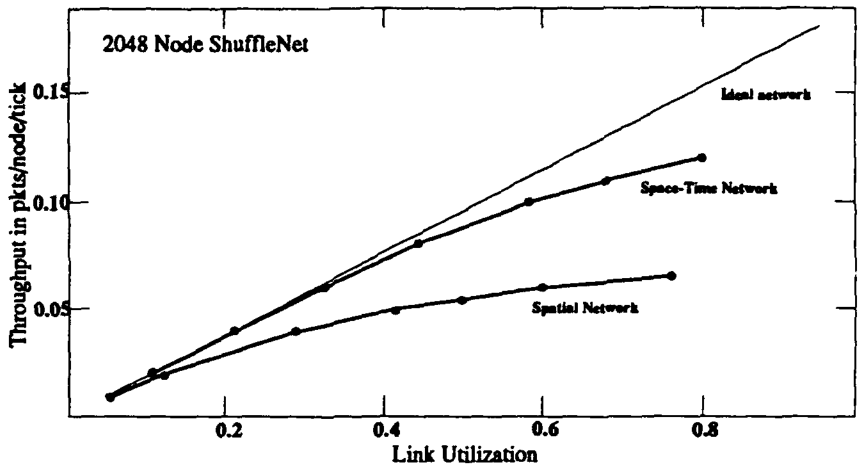


Figure 5.11: Throughput vs. Link Utilization for a 2048 Node ShuffleNet

Though the tail is shortened, the average behavior of the network does not change significantly. Figures (5.12) and (5.13) show the latency distribution histogram for a 2048-node ShuffleNet with 2×2 nodes and 2S2T nodes under the same injection rate and age-priority scheme outlined above. In each case, the ideal distribution, under no deflections, has also been plotted for comparison. Apart from reducing the average latency, the space-time network also shortens the tail of the latency distribution, i.e. it reduces the variations among the latency of packets. For the same offered load, as was noted earlier, the reduction in the number of deflections suffered by packets in a space-time network leads to lower average latency and lower link utilization. The reduction in the tail of the distribution of Fig. (5.13) is not because of the reduction in link utilization from 0.5 in the spatial network to 0.33 in the space-time network. Figure (5.14) shows that even at twice the load offered earlier, the spread in latency under space-time switching at nodes is smaller than the spread seen in Fig. (5.12). The link utilization in this case is 0.8.

Thus the performance of deflection routing networks can be greatly

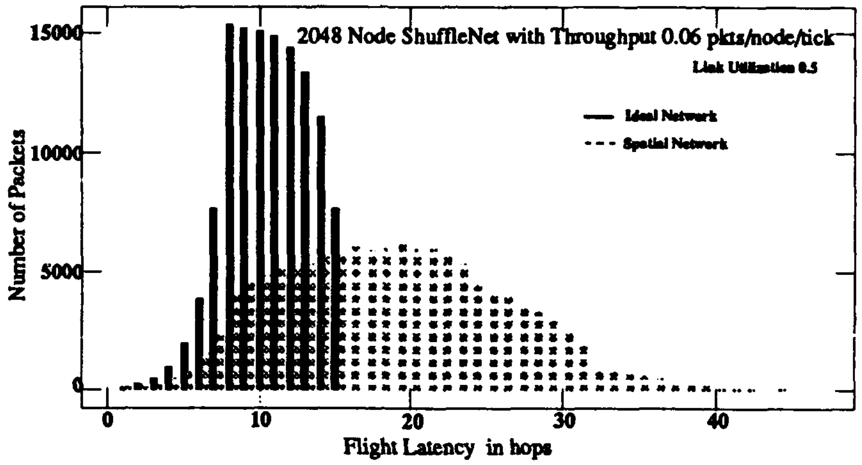


Figure 5.12. Flight Latency Histogram for a 2048 Node ShuffleNet with 2X2 Nodes

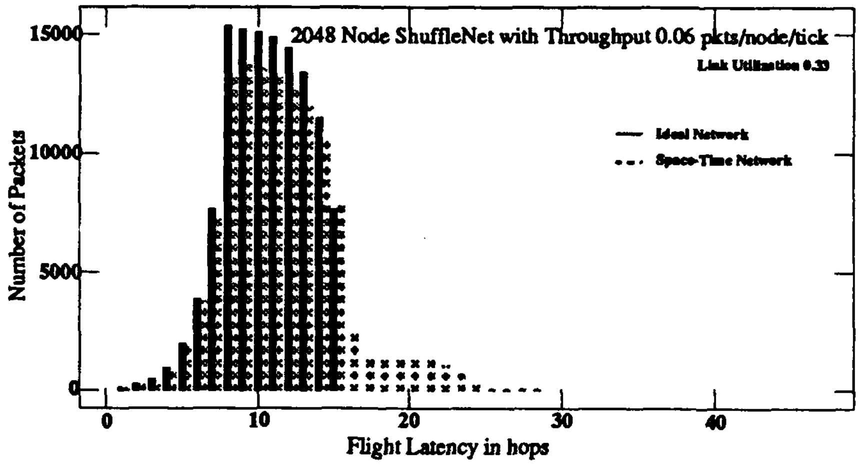


Figure 5.13. Flight Latency Histogram for a 2048 Node ShuffleNet with 2S2T Nodes

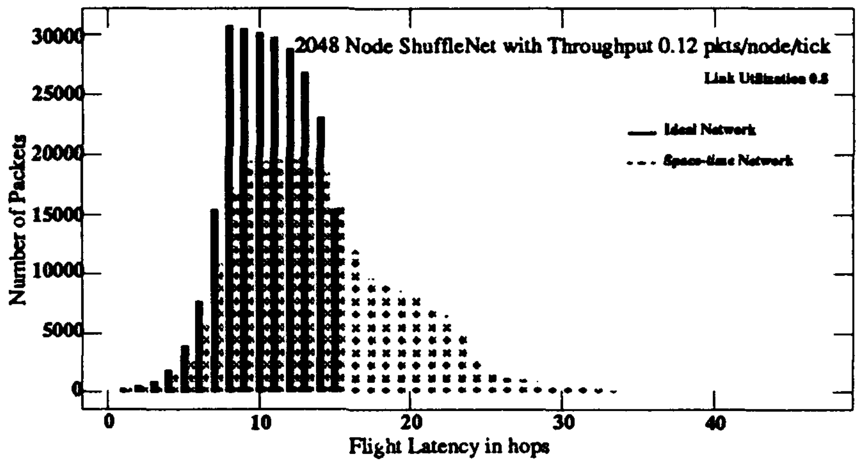


Figure 5.14. Flight Latency Histogram for a 2048 Node ShuffleNet with 2S2T Nodes

enhanced by combining time-domain switching with spatial switching at the nodes. Under uniform loading conditions, the behavior is very close to an ideal network up to moderate link utilizations as shown by Figs. (5.8) through (5.11). Though there is a deviation from the ideal at high link utilizations, the space-time technique still exhibits superior performance. Figures (5.15) and (5.16) show the variations in average flight latency and maximum node throughput as a function of network size. These maximum values were obtained through solution of the ShuffleNet model presented in section 3.2. Though the flight latency in a space-time network is higher than the ideal, the differential does not increase noticeably with network size. The reduction in network latency leads to consistent improvement in user throughput.

5.4 Simulation Results on MSNet with 2S2T Nodes

In this section we present the results on the simulation of a MSNet under space-time switching. The network was simulated under uniform loading conditions. We present results for 64 node (8×8) node networks. Figure (5.17)

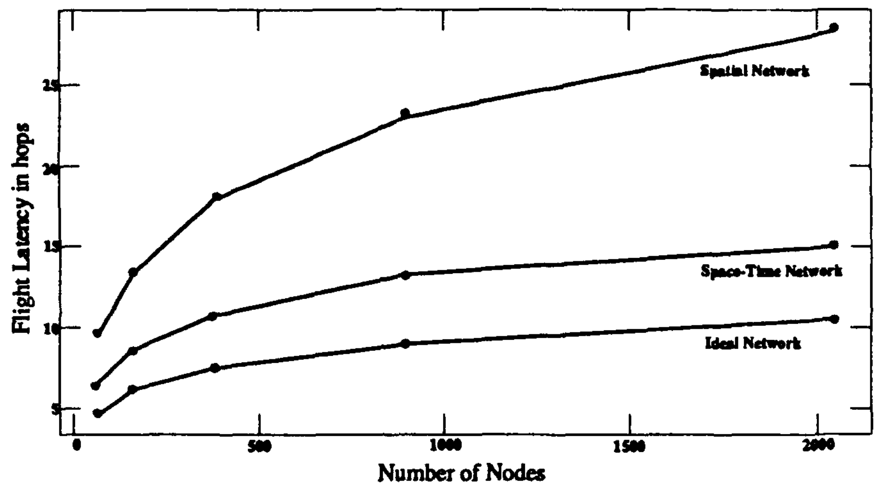


Figure 5.15: Variation of Average Flight Latency with Size

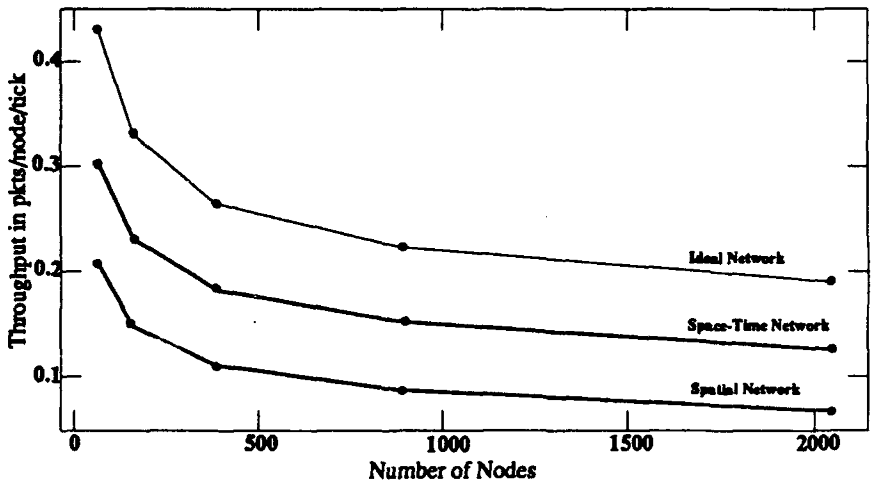


Figure 5.16: Variation of Maximum User Throughput with Size

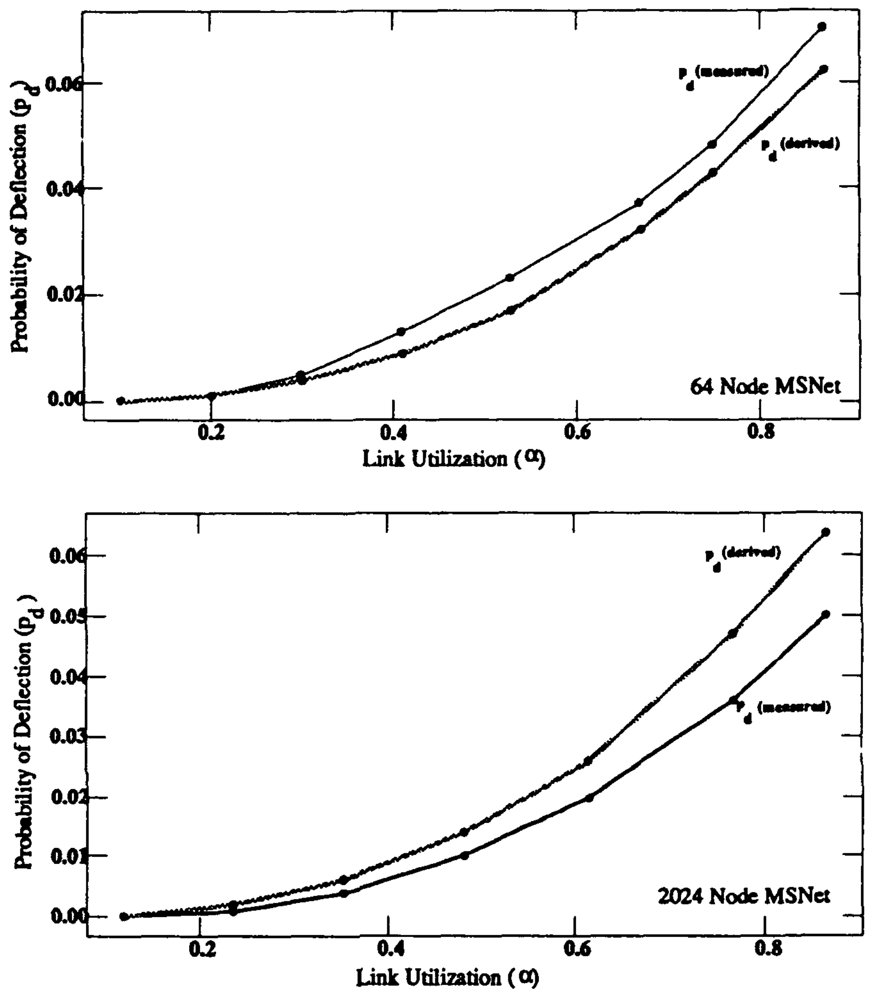


Figure 5.17: Probability of Deflection of a Packet inside a MSNet

shows the measured probability cf deflection and the probability curve obtained from the values derived by substituting the measured values of link utilization and care probability in Eqn. (5.2). The derived values are slightly lower for the 64 node network as was observed in the case of the ShuffleNet. For the 2024 node network the derived values are slightly higher. A 400 node (20×20) network was also simulated. The results (not shown here) on the measured and derived values were very close to each other. This reversal in trend may

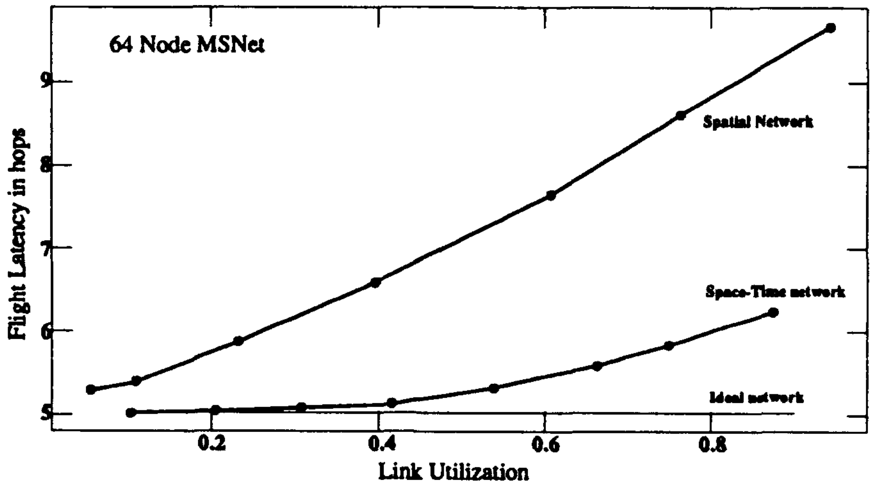


Figure 5.18: Flight Latency vs. Link Utilization for a 64 Node MSNet

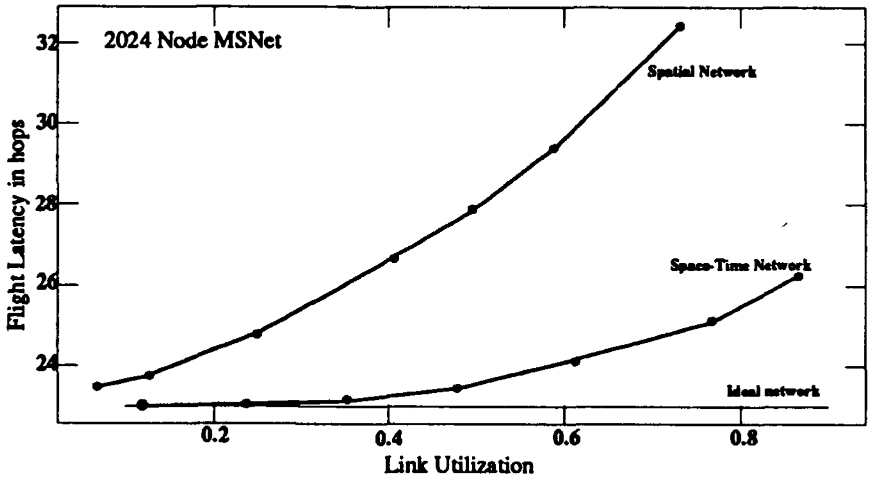


Figure 5.19: Flight Latency vs. Link Utilization for a 2024 Node MSNet

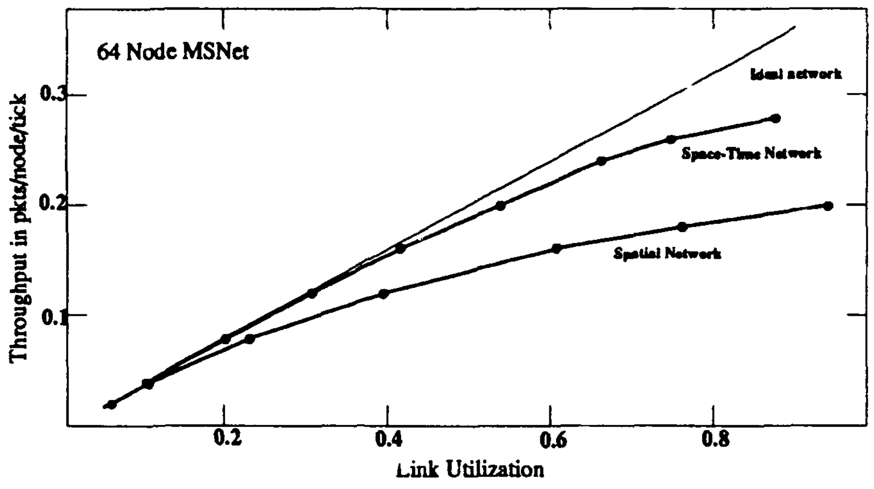


Figure 5.20: Throughput vs. Link Utilization for a 64 Node MSNet

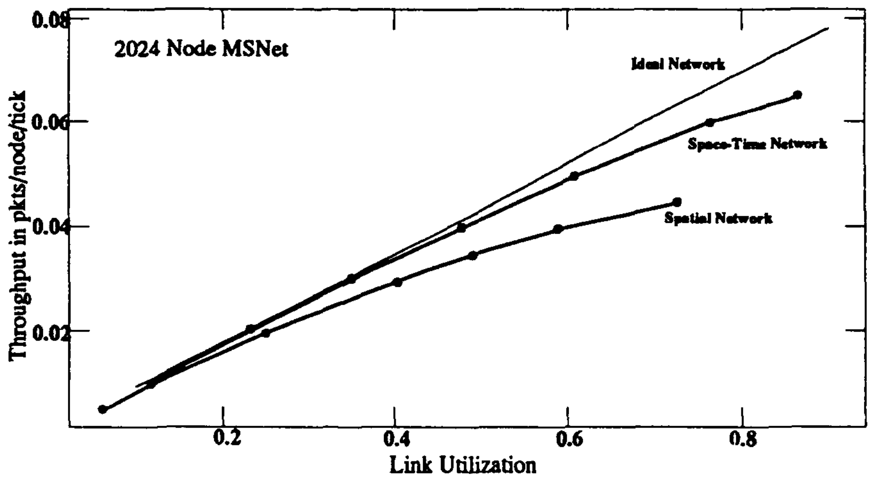


Figure 5.21: Throughput vs. Link Utilization for a 2024 Node MSNet

be due to the interaction between the size and the simplified, slightly sub-optimal routing algorithm used. The effect of the routing algorithm is larger fluctuations in instantaneous link utilization.

The flight latency behavior with respect to the link utilization for the 64 node and 2024 node networks is shown in Figs. (5.18) and (5.19). A comparison of these figures with Figs. (5.8) and (5.9) for the ShuffleNet shows that space-time switching has about the same effect in reducing the latency in a ShuffleNet as the MSNet. The corresponding throughput behavior of MSNet is shown in Figs. (5.20) and (5.21). While the behavior of 64 node networks of the two topologies is about the same, there is a marked difference in the effect of space-time switching on the bigger network. A comparison of Fig. (5.11) with Fig. (5.21) shows that the improvement in throughput of the 2048 node ShuffleNet is much more prominent than the improvement in the throughput found in the 2024 node MSNet.

Figures (5.22) and (5.23) show a flight latency distribution in a 2024 node MSNet with 2×2 nodes and 2S2T nodes respectively. In both cases the offered load or throughput was 0.04 pkts/node/tick, which led to a link utilization of 0.6 in the case of the 2×2 node configuration and a link utilization of 0.5 in the case of 2S2T node configuration. As was seen earlier in the case of the 2048 node ShuffleNet (Figs. (5.12) and (5.13)), the 2S2T configuration leads to a distribution that is very close to the ideal for the network.

5.5 General behavior of Space-Time Networks

Simultaneous space and time switching makes packet-switched direct networks nearly as efficient, with reasonable network parameters, as an ideal network without contention. Limited temporal reordering greatly reduces the

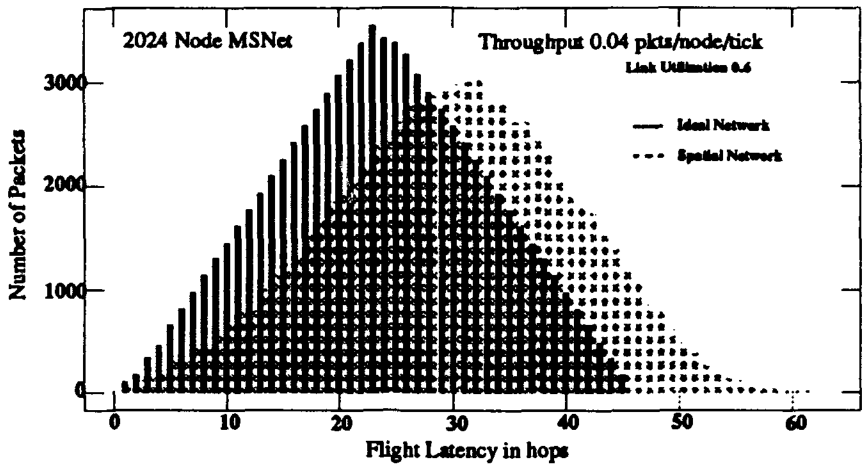


Figure 5.22. Flight Latency Histogram for a 2024 Node MSNet with 2X2 Nodes

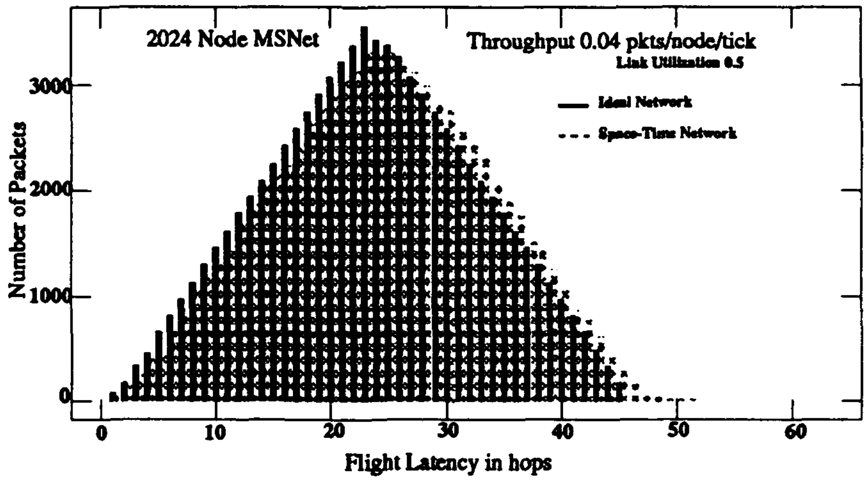


Figure 5.23. Flight Latency Histogram for a 2024 Node MSNet with 2S2T Nodes

contention for output spatial channels at the switching points. In conventional packet switched networks under a deflection routing protocol, packets are stored in flight in the network links rather than being buffered statically at the switching point. The space-time switching nodes cause contending packets that would otherwise encounter deflection to be reordered in time if the contention can thus be removed. Effectively, the dynamic buffering function of the network is reduced. The space-time switching paradigm may appear to be similar to finite buffering in store-and-forward networks. The most important feature that distinguishes the two is that time is brought into the switching paradigm explicitly in a space-time node. The two time channel sliding window node we have analyzed allows packets arriving simultaneously to be switched along with packets that arrived in the previous network cycle as well as the packets that arrive in the next network cycle, thus providing a node with a dynamic buffering capability that is more than what single buffers at the outputs can provide. Due to lack of static buffers in optical implementations, dynamic output buffering might be provided at each output channel. However, considerably less hardware is required to implement simultaneous space and time switching.

Since the required switching logic is simple and pipelined, the network links can be very fast. As mentioned in Section 4.1, a time-slot interchange involves a frame delay as well as additional switching delay. These delays will add a few pipeline stages to each switch node without altering the link bandwidth. We find that the increase in latency due additional delay at 2S2T switch nodes is more than offset by the decrease in latency due to the reduced number of deflections even at short internode distances, while maintaining

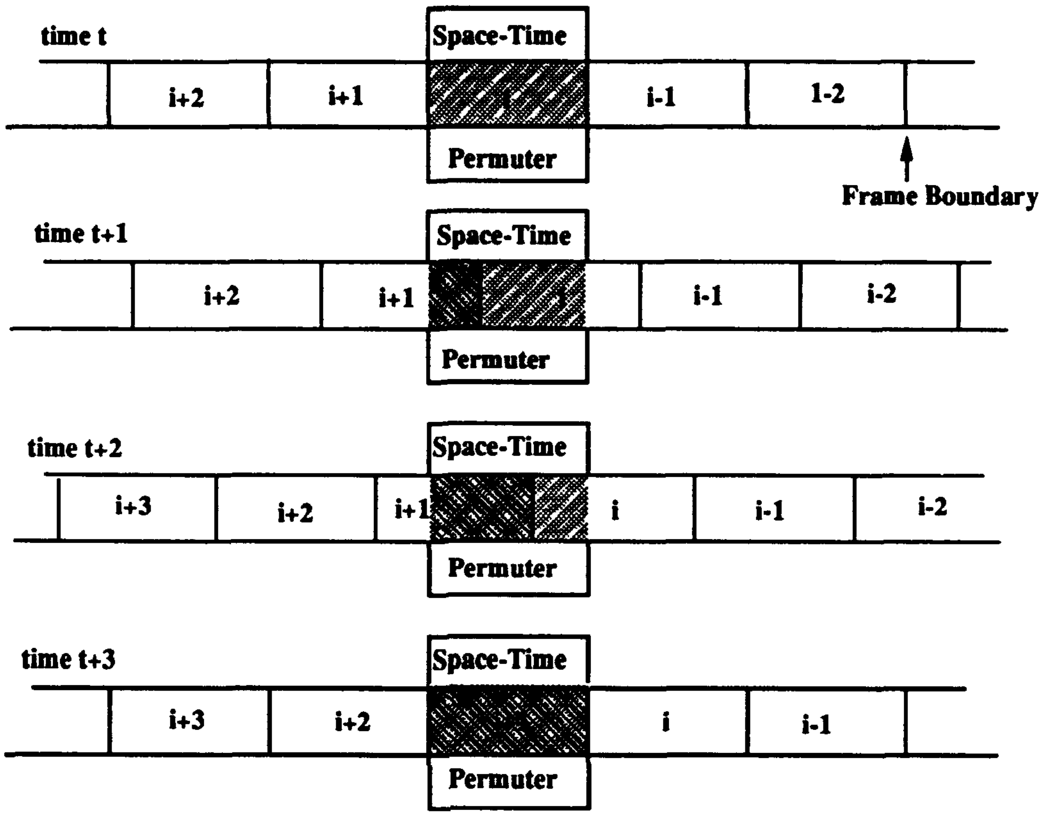


Figure 5.24. A schematic representation of a Space-Time Permuter with permutation window fixed on blocks

the advantage of increased throughput. This effect will be seen in the results presented in the next chapter.

5.6 Classification of Space-Time Switching Techniques

Space-time switching is a general concept which involves spatial rearrangement and temporal reordering of inputs to produce the desired output. While the approach presented in this chapter is particularly suitable for fast packet-switched computer interconnects, other approaches are possible [17, 18, 74]. Two orthogonal issues can be identified to classify the different approaches. One of them concerns the selection of the time window and the other concerns the relationship between the input and output windows.

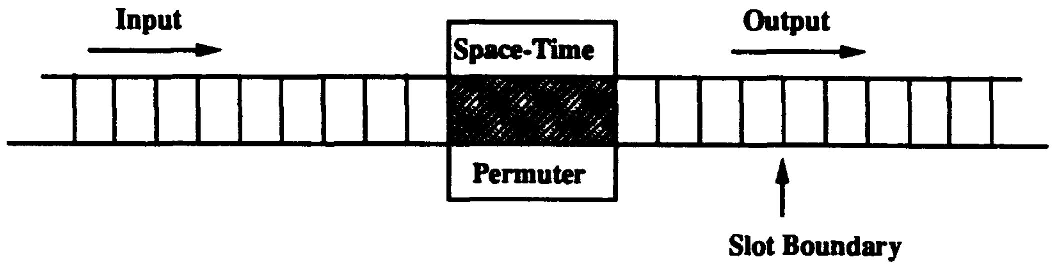


Figure 5.25. A schematic representation of a Space-Time Permuter with sliding permutation window

The window on input slots could be a train of successive blocks of slots in space and time (Fig. (5.24)) or a sliding window on a block of slots that are within a permutation boundary fixed with respect the node as shown in Fig. (5.25). In Fig. (5.24) we consider blocks 3 time-slots wide. At some time t measured in network cycles, block i fully appears inside the permutation window of the space-time permuter. The switch setting for the necessary permutation are determined and used during times t, t+1, t+2. During times t+1 and t+2 parts of block i will move out of the permutation window, while block i+1 will start to move in. During time t+3 the entire block i+1 will enter the permutation window and its permutation will be computed and effected during times t+3, t+4 and t+5. The fixed window approach allows rearrangement of slots within each successive block with no slot moving between blocks. The sliding window on the other hand, allows restricted continuous rearrangement of slots as they pass through the node. During every network cycle, one slot will move out of the permutation window at the output and another slot will enter at the input. Permutations are computed continuously for the slots inside the permutation window.

There are two possible modes of operation of the node that lead

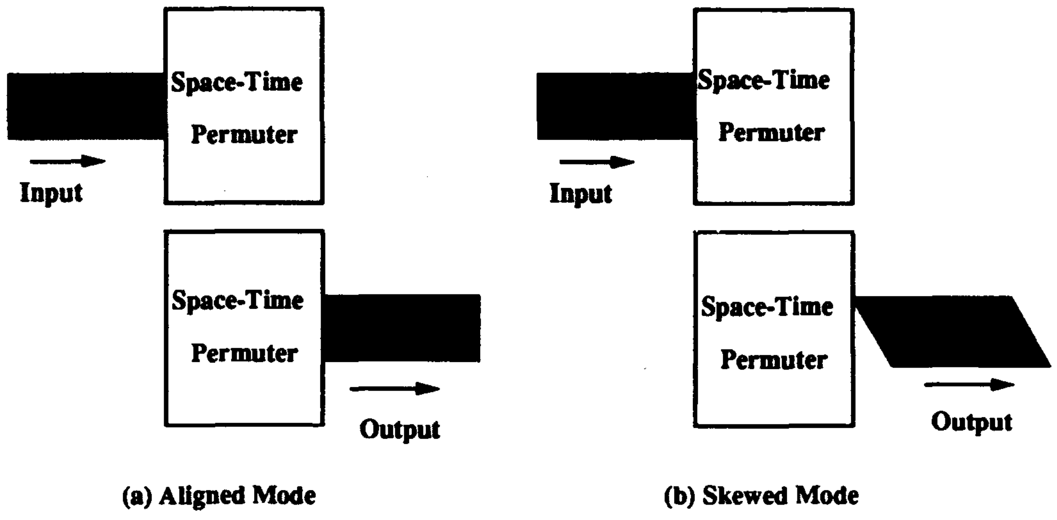


Figure 5.26. Two different switching modes possible in a Space-Time Permuter

to a difference in the relationship between an input window and the corresponding output window. Under aligned switching mode, packets that enter simultaneously and which do not require a space-time permutation exit the node at the same time. Under skewed switching mode, non permuted simultaneous packets exit the node with their relative positions skewed in time. Figure (5.26) illustrates the two switching modes. Traditional time-slot interchangers [86, 15, 17, 18] fall in the fixed window, aligned access category. The space-time switch proposed in this work falls in the sliding window, aligned access category. While there seem to be no specific example that uses the fixed window under aligned mode, Chlamtac *et.al.* [74] consider a sliding window under skewed switching mode. As illustrated by the example of Fig. (5.1), a sliding time window provides the node with more contention resolution power than a fixed time window and hence, is more suitable for packet-switched networks under deflection routing. While we arrived at the sliding window concept as well as the aligned switching mode based on this consideration, Chlamtac *et.al.* have independently arrived at a similar solution using skewed switching mode from dynamic delay line buffering considerations.

The aligned switching mode provides equal internode transit time for simultaneous packets that are not permuted in time, and the control can be computed faster by the routing control processor as compared to the skewed switching mode. For the same hardware in the space-time permuter, i.e. 3 switches and 2 delay lines, the skewed access mode can provide a slightly better performance. The choice between the switching modes, depends on the intended application. Multiprocessor interconnects carry short packets, and hence require the switch settings to be computed within a packet slot duration, typically *few ns* and hence, our aligned access node is preferable. In contrast, communication networks carry long packets and require new switch control setting to be generated once in about 100*ns*. So, they can afford increased complexity in the control algorithm needed by the skewed access mode.

CHAPTER 6

THE MULTIPROCESSOR PERFORMANCE

6.1 The Multiprocessor Model

The study of the network behavior presented in the last two chapters was aimed at understanding the behavior of direct networks that have a network link bandwidth that is far greater than the insertion bandwidth of the hosts. The analytical as well as simulation studies were performed assuming uniform workloads, with messages originating from a source and terminating at their respective destinations. This chapter presents a simulation based analysis of a model multiprocessor system interconnected by the ultrafast networks that have been analyzed so far. As mentioned in Chapter 1, the network link bandwidth in existing shared memory multiprocessor systems is comparable to the insertion bandwidth of the processors. On the contrary, the ultrafast networks under study have a high bandwidth differential between what is needed by the hosts and the link bandwidth that the network possesses. Also, the topology of the networks lead to non-uniform memory access times. The model system considered here is an highly abstracted model of a multiprocessor system. Basically each node in the network has a host consisting of a processor-memory pair attached to it. Neither the processors nor the memories are simulated fully. A number of simplifying assumptions have been made to simulate the system, while maintaining the important features intact.

The first step towards building a meaningful model for any system is to develop a detailed description of the system so that the parameters that are required for modeling of the system can be abstracted. The next step is to define the workload under which the system will be evaluated. Another important aspect is to define the type of information that should be obtained through model solution, so that the characteristics of the underlying system can be extracted. In this section we present a detailed system description, the workload presented and the metrics used for monitoring and assessing the performance of the system.

6.1.1 System Description. Our aim is to study a large shared memory multiprocessor system, consisting of hundreds of processors, interconnected by an ultrafast network. The possibility of exploiting high link bandwidth provided by all-optical or optoelectronic interconnects imposes certain assumptions about system specifications. Since memory and logic cycles are expensive resources at rates commensurate with link bandwidth in such interconnects, the system needs a flow-through architecture with no static buffering at switching nodes. So, we restrict our attention to direct network topologies under a hot-potato/deflection routing protocol. At the top level of abstraction, the system has the following features:

- The system is a distributed shared memory system with processor-memory pairs interconnected by a pipelined interconnection system. A host consisting of a processor-memory combination is attached to each node in the network.
- The program memory and data memory are totally independent of each other. While the program memory and private part of data memory reside totally in the processor's local memory, the shared memory is distributed

among all the local memories. Only the shared data memory is accessed by the processors through the interconnection network.

- The global address space is uniformly shared by all processors in the system. The memory requests are satisfied directly by the memory modules, i.e. without intervention from the accompanying processor.
- The mean time between access by the processors is significantly higher than the network cycle time, leading to the ultrafast operation region.
- Packet switching of messages is assumed. Every request presented to the network is packaged into a single message packet and is switched from one stage to the next by the pipelined switch nodes.

The specific characteristics of the processors and memories are:

- The processors run multiple threads and are capable of context switching in few cycles.
- The processors are pipelined such that a memory access request can be issued by each processor every clock cycle.
- The memory is fully pipelined at the same rate as the processors. Because the memory is pipelined, memory access latency is fixed.

The following assumptions have been made to specify the characteristics of the interconnection.

- The network is assumed to be fully synchronous with a flow-through architecture. Packets keep moving through the network until they reach their destination. The switch nodes adopt the deflection routing protocol. They contain no buffers. They hold packets only for the duration of time needed to make the routing decision.

- The switch nodes in the network are fully pipelined so that they can accept a packet through each input port every network cycle. Packets injected by a host will be accepted by the associated switch node if an incoming slot is empty or if an incoming packet is destined for the host.
- Two regular, but widely different network topologies are considered. One is the logarithmic ShuffleNet and the other is the planar toroidal MSNet. These two topologies represent two extremes in diameter versus deflection penalty tradeoff. Also, the ShuffleNet is a logarithmic topology which has a predictable behavior with respect to deflections, i.e. every deflection takes it around all the columns one more time provided there are no further deflections. The movements of packets in a Manhattan mesh is not as predictable, which makes it difficult to obtain a closed form expression for the flight latency even under an uniform load.
- The switches have an external in-degree and out-degree of two into the network. Each switch node can receive two packets from the two nodes connected to its input and send two packets out through links to nodes connected to its output every network cycle.
- A host can inject at most two messages into the network during any given cycle provided the attached switch node can accept the injected packets.
- The switch nodes are self-routing. All information needed to route packets is present in the packet headers. The routing processors at the switch nodes make routing decisions based on the packet header information.

Figure (6.1) is a schematic block diagram of the network interface unit that shows its internal structure and its connections to the host processor, the host memory and the attached network node. The buffer structure is

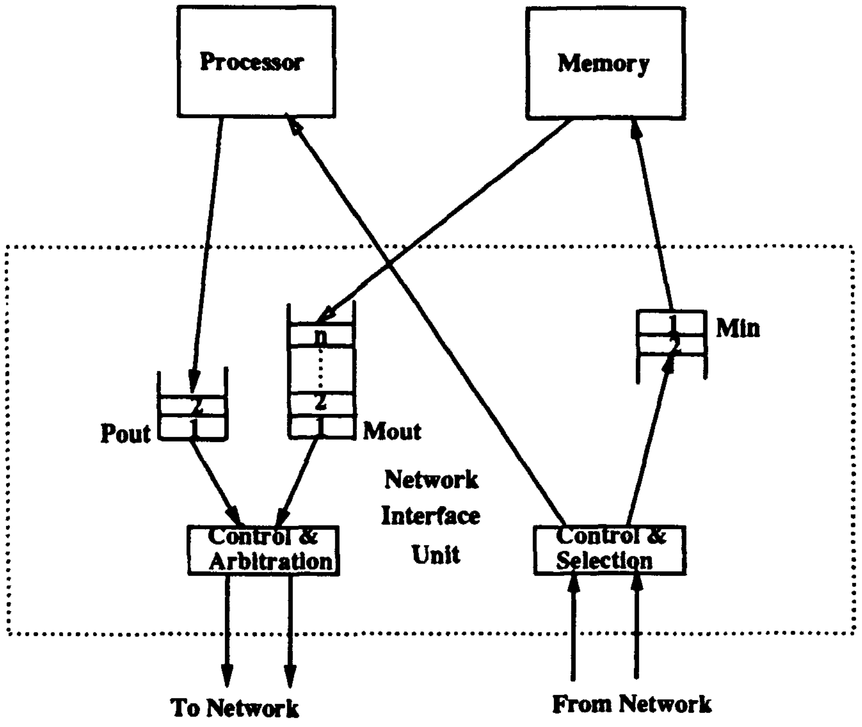


Figure 6.1. A schematic diagram of the organization of the Network Interface Unit.

designed the way it is due to the assumptions that have been made. We have assumed that enough resources are available to send a reply packet that has reached its destination node into the associated processor. So, we do not have a buffer in the Network Interface Unit that holds packets that need to go to the host processor. We provide a buffer of size two (P_{out}) between the processor and the arbitration unit that sends the packets into the network for buffering requests originating at the processor that need to access the network. During any given cycle, the network can accept at most two requests. So, we allow for the possibility that given the network can accept two packets and that the memory has no packet to inject, the processor will be able to inject two packets. When buffer P_{out} is full, the processor is blocked from generating any new request. The buffer to hold packets coming into the network to access the memory is also two packets deep. When the buffer slots in M_{in} are empty and two packets requiring access to the memory module come in simultaneously on the two links to the attached network node, both can be accepted. Under uniform loading conditions it is highly unlikely, that both incoming packet slots on the network links will have packets requiring access to the memory on two consecutive network cycles. So, the two packet deep buffer should be sufficient. The M_{out} holds packets that emerge from the memory pipeline and need to go back to the source of the request. It is possible that such packets may have to wait for a non-deterministic amount of time to enter the network. This buffer has a large capacity, so that it will not be necessary to block the memory pipeline, because the buffer is full. The control & arbitration unit has two functions. The arbitration function helps to provide packets from both the processor and the memory a fair access to the network. The control function helps to block the processor when P_{out} is full. The control & selection unit also has two functions. Its control function is to send a signal to the attached network node indicating how many memory access packets it can receive. The selection function separates the packets that need to be routed to the processor from those to the memory, sends the packets destined to the processor straight through and puts the packets needing memory access in M_{in}.

The characteristics of the multiprocessor system outlined above were further abstracted, and some simplifying assumptions were made to carry out the simulations within the framework of these specifications. While the functionality of the interconnect was fully simulated, the host modules were highly simplified. The only processor function modeled is the issue and receipt of remote shared memory requests. Though the system is multithreaded, no account of distinct threads is maintained. It is assumed that enough threads are available so that the processor can be kept busy all the time. The memory is implemented as a simple pipeline which can satisfy requests at the rate of one per tick or cycle, but may have latency of more than one tick. A processor's access to shared data items in the memory module associated with it is assumed to be through its own channel for private data and is not modeled explicitly. Possible conflicts at memory modules between accesses by local and remote processors is not modeled. In summary, the processors submit a memory access request to remote locations through the network at times determined by the workload presented. No distinction is made between reads, writes, and locks or any other synchronization primitive. Even with this extremely simplified approach, as will be shown in the following sections, it has been possible to extract some important characteristics of ultrafast multiprocessor networks.

6.1.2 The Workload Model. The workload used in simulation of systems can be either stochastic or deterministic. In a stochastic workload model, the next service request is selected using generating functions based on random deviates. Such workloads tend to capture the essence of system operations by using appropriate probability distributions. Stochastic workloads offer the advantage that they can yield a functional relationship between the input and result parameters. The alternative to probabilistic workload is deterministic workload provided through processor execution traces. Its principal advantage is a full representation of a specific job, provided an exact knowledge of the simulated system's workload is available. The main drawback of the trace driven model of workload characterization is that the results are limited to the specific case under test, and it is difficult to quantify the performance metrics and relate them to the input parameters.

Since this study is aimed at gaining a fundamental insight into the behavior of ultrafast interconnects, a stochastic workload model has been used. The workload needs to be characterized both in space and time. The request rate of the processor relates to the temporal part of the workload. Apart from the requirements of the application program, it will also be influenced by the number of outstanding requests from the processor. The spatial aspect of workload concerns the address distribution of the requests, which is determined by the application program. The complexity of the stochastic model can be varied according to the level of abstraction desired. For example, one could study the effects of high level shared memory program constructs such as barriers, critical sections, etc. It is also possible to create a composite workload, an example of which is the replicated workload model used in [87]. In this model, the same synthetic workload is presented to all the processors. The generic program is an iterative loop which has two regions. The first region has an intermix of local and shared memory accesses. The second region is a critical section protected by a lock-unlock operation. The workload used in this study is even more abstracted. The system performance is first analyzed under uniform traffic conditions. Then, workloads with only spatial or temporal non-uniformities are considered. All real workloads are a complex intermix of such spatial and temporal non-uniformities. Though it is not possible to extend the findings based on simple behavior easily to complex situations, the simple analysis can provide an insight into the general system behavior. The three kinds of workloads that were used in the simulations are described below.

- While studying uniform traffic, no limit was placed on the number of outstanding requests from a processor. Processors continuously generated requests at a constant rate. The rate was varied in steps until the network reached saturation.
- The spatial locality was studied by the traditional method that has been used in the literature [88, 89] to study *hot-spot* behavior in indirect networks for multiprocessors. All processors generate requests at the same uniform rate. A small fraction of the requests from each processor is directed to one hot memory module, while the rest of the requests is distributed uniformly among all memory modules.
- In order to separate the temporal behavior from the spatial behavior, temporal locality has been studied by turning all processors into the burst mode at the same time. Bursts of constant size are generated after the system is initialized into a steady state with a uniform load. Time to relax back to the condition that existed before the burst is noted along with other usual performance parameters.

A more detailed description of the workload is presented at the beginning of the relevant sections presenting the results.

6.1.3 Performance Metrics. The node hosts, i.e. the processors and memories, use the interconnection network to exchange packets of information. In Section (3.1.2) a number of performance metrics related to the network behavior were defined and used. Some of those metrics, such as Flight Latency and Link Utilization, will be used here also. We need to define a few more parameters that are specific to the system description. The new metrics are:

Roundtrip Latency Roundtrip latency is the time between placement of a request by the processor into the network interface unit and the time the processor receives the reply from the remote location. The round trip latency has a number of contributions including the flight latency.

Memory Access Bandwidth It is the rate at which the processors access the memory through the network. The inverse of memory access bandwidth is the time between two requests placed by a processor including any overheads for context switching between threads.

Mean T'me between Access It is the average time between two remote shared memory accesses issued by a process or thread. This quantity is dependent on the processor architecture as well as the application program running on it.

Number of Threads It is the minimum number of threads needed to keep the processor busy under normal operating conditions. By normal operating conditions, we mean conditions that don't deviate the system far from its average behavior.

6.2 Behavior under Uniform Load

This section concerns the performance of the model system under uniform load. A uniform load was created by allowing each processor to generate packets randomly at a uniform rate, independent of other processors. The address distribution among the packets was also randomized uniformly, so that the requests and replies flowing through the network created a uniform load on it. The simulations were carried out for both the spatial and space-time configurations with ShuffleNet and MSNet topologies. Two different sizes were simulated for each topology. Systems with ShuffleNet topologies had 384

($k = 6$) and 896($k = 7$) nodes. 20×20 (400 nodes) and 30×30 (900 nodes) MSNets were simulated to provide a comparison between networks of nearly the same size with the two topologies under investigation. The results of the simulations are summarized in Tables (6.1) through (6.8). 95% confidence intervals are shown for all latencies. The entries marked by an asterisk in the link utilization column denotes observations in which the system did not settle into a steady state. The knee of the throughput round trip latency curve was reached somewhere between that observation and the previous one. In order to simulate the behavior of the system under offered load, it is necessary to specify the internode link length. In all simulations we assume that the switch node has 5 pipeline stages and the link between nodes is 5 ticks long, leading to an internode distance n_l of 10 ticks for a purely spatial network. The introduction of frame delay overhead of 1 tick in the space-time switch leads to an effective internode distance of 11 ticks in the space-time configurations.

The round trip latency of a packet is the sum of several components. The major components are the flight latencies to and from the remote memory module. The other components are the time to access the memory, and the time spent by the packet in the network interface unit(NIU). The memory access latency and the NIU latency both have fixed and variable subcomponents. The fixed parts are the times required to carry out the functions required of them. The variable parts are the wait times spent at the memory to gain access to memory and at the NIU to find a free network slot to enter the network. These variable contributions to latency are dependent on the volume and nature of the load on the system. The memory was assumed to have 4 pipeline stages, and hence the fixed part of memory access latency is 4 ticks. It was assumed

Table 6.1. Flight and Roundtrip Latencies in the 384 Node ShuffleNet Configuration with Spatial Nodes

Mem. Acc. BW η (pkts/node/tick)	Flight Latency (ticks)	Link Utilization	Roundtrip Latency (ticks)
0.01	81.1±0.6	0.08	168.5±1.2
0.02	88.5±0.5	0.18	183.0±1.1
0.03	97.6±0.5	0.29	201.5±1.0
0.04	110.7±0.5	0.44	228.0±1.1
0.05	131.6±0.6	0.66	270.9±1.1
0.055	146.7±0.6	0.80	301.4±1.2
0.06	161.7±0.6	0.97*	339.7±1.2

Table 6.2. Flight and Roundtrip Latencies in the 384 Node ShuffleNet Configuration with Space-Time Nodes

Mem. Acc. BW η (pkts/node/tick)	Flight Latency (ticks)	Link Utilization	Roundtrip Latency (ticks)
0.02	82.8±0.4	0.15	171.5±0.8
0.04	84.3±0.3	0.31	175.0±0.6
0.06	87.8±0.3	0.48	182.1±0.6
0.08	95.3±0.3	0.69	198.4±0.6
0.09	102.5±0.3	0.84	214.8±0.6
0.10	107.1±0.3	0.97*	228.2±0.7

that the NIU takes 1 tick to package the packet to be sent into the network. Hence, the fixed part of NIU latency is 2 ticks, 1 for the forward message and 1 for the reply from memory. So, any difference between the average round trip latency and the sum of twice the average flight latency, the fixed memory access latency and the fixed NIU latency is due to wait times at various buffers in the hosts.

We have assumed that processors generate a request statistically once every $1/\eta$ ticks. In the steady state, they will receive a reply at the same rate. Varying η, while maintaining the network under steady state, leads to different

Table 6.3. Flight and Roundtrip Latencies in the 400 Node MSNet Configuration with Spatial Nodes

Mem. Acc. BW η (pkts/node/tick)	Flight Latency (ticks)	Link Utilization	Roundtrip Latency (ticks)
0.01	116.3±1.0	0.12	237.8±2.6
0.02	125.6±0.7	0.25	257.3±2.0
0.03	136.2±0.7	0.41	278.4±1.7
0.035	145.0±0.7	0.51	296.8±1.6
0.04	159.6±0.7	0.64	326.5±1.6
0.045	173.5±0.7	0.78	356.4±1.6
0.05	193.2±0.7	0.97*	402.4±1.6

Table 6.4. Flight and Roundtrip Latencies in the 400 Node MSNet Configuration with Space-Time Nodes

Mem. Acc. BW η (pkts/node/tick)	Flight Latency (ticks)	Link Utilization	Roundtrip Latency (ticks)
0.01	119.8±1.0	0.11	245.4±2.8
0.02	121.8±0.7	0.22	249.7±2.0
0.04	125.3±0.5	0.46	257.2±1.4
0.05	129.7±0.5	0.59	266.3±1.3
0.06	137.0±0.5	0.75	282.0±1.3
0.065	142.4±0.5	0.84	293.1±1.3
0.07	148.0±0.5	0.94*	310.1±1.3

Table 6.5. Flight and Roundtrip Latencies in the 896 Node ShuffleNet Configuration with Spatial Nodes

Mem. Acc. BW η (pkts/node/tick)	Flight Latency (ticks)	Link Utilization	Roundtrip Latency (ticks)
0.01	99.8±0.5	0.10	205.3±1.0
0.02	113.9±0.5	0.23	234.1±0.9
0.03	134.2±0.5	0.40	274.7±0.9
0.035	150.0±0.5	0.53	306.6±1.0
0.04	172.2±0.5	0.69	351.7±1.0
0.045	202.0±0.6	0.91*	415.5±1.1

Table 6.6. Flight and Roundtrip Latencies in the 896 Node ShuffleNet Configuration with Space-Time Nodes

Mem. Acc. BW η (pkts/node/tick)	Flight Latency (ticks)	Link Utilization	Roundtrip Latency (ticks)
0.02	99.8±0.3	0.18	205.7±0.6
0.04	102.5±0.2	0.37	211.3±0.5
0.06	111.3±0.2	0.61	229.8±0.5
0.07	120.5±0.3	0.77	250.0±0.5
0.075	127.7±0.3	0.87	266.5±0.6
.08	132.6±0.3	0.96*	279.9±0.6

Table 6.7. Flight and Roundtrip Latencies in the 900 Node MSNet Configuration with Spatial Nodes

Mem. Acc. BW η (pkts/node/tick)	Flight Latency (ticks)	Link Utilization	Roundtrip Latency (ticks)
0.01	170.0±1.0	0.17	346.2±2.6
0.02	187.7±0.7	0.38	381.1±1.9
0.025	200.9±0.7	0.50	409.1±1.8
0.03	221.9±0.7	0.67	451.3±1.7
0.032	233.1±0.7	0.75	475.1±1.7
0.035	256.2±0.7	0.90*	526.0±1.7

Table 6.8. Flight and Roundtrip Latencies in the 900 Node MSNet Configuration with Space-Time Nodes

Mem. Acc. BW η (pkts/node/tick)	Flight Latency (ticks)	Link Utilization	Roundtrip Latency (ticks)
0.01	175.3±1.0	0.16	356.6±2.8
0.02	177.9±0.7	0.32	361.8±2.0
0.03	182.8±0.6	0.50	372.0±1.6
0.04	191.2±0.5	0.70	390.3±1.5
0.045	200.6±0.5	0.82	410.3±1.4
0.05	212.3± 0.5	0.97*	441.3±1.4

link utilizations. The maximum possible memory access bandwidth, $1/\eta_{max}$, available to the processors determines how often they can access the shared memory for a particular configuration of the system. In coarse grained multithreaded processors, a resident context continues to execute until it hits a remote access. The overhead for a context switch is typically a few processor cycles. The mean time between two shared memory accesses by a processor, t_{access}, is not the same as $1/\eta_{max}$ because of the finite overhead due to context switching. In our model, we have assumed that all instructions and private data are local to the processor and only shared memory is remotely accessed through the network. In general, a processor need not wait for a write request to be acknowledged before continuing execution, but it has to wait for return of requested data to continue execution. The fraction of shared memory requests on which a process has to wait depends on the application running on the system. In this analysis we parameterize the fraction by σ. Under the assumptions made, t_{access} is given by

$$t_{access} = 1/\eta_{max} - \sigma C. \tag{6.1}$$

where C is the time taken by the processor to switch context. Figures (6.2) through (6.5) show a plot of the minimum mean time between accesses that is possible as a function of context switch overhead obtained by solving Eqn. (6.1) using values of η_{max} obtained through simulations of the different configurations. For each system configuration, three values of σ are considered. σ=1 denotes a situation when a thread waits on all replies from the remote memory. In practice, processors will context switch only when they need a data item from the remote memory. σ=0.5 denotes a situation when a thread waits, on an average, on every other remote memory access. For example, this situation will arise under the naive condition that the only possible memory accesses are unconditional reads and writes and that the number of reads and writes are balanced. Results for an intermediate value of σ=0.67, which provides a ratio of 2 to 1 between requests that need to be satisfied before the process can continue and requests that do not affect the continuation of the process, is also shown. If the actual mean time between accesses is greater than the minimum value required according to these plots, then the access bandwidth required is less than what the network can provide. Under such conditions, the network can satisfy the system requirements without any bottleneck at any point in the system. The figures reveal that a shared memory access rate of 1 in 10 cycles or ticks can be easily supported by the 384 node ShuffleNet system with space-time nodes, and with comparable context switch overhead by the 400 node MSNet and 896 node ShuffleNet configurations with space-time switching at nodes. All other configurations will require a context switch time far more than the mean time between accesses by the processor to satisfy the communication requirement. Figures (6.6) and (6.7) show predicted minimum time between accesses that is possible in a 2048 node system with ShuffleNet topology and a 2024 node system with MSNet topology. The predictions are based on the maximum memory access bandwidth derived from the simulation results presented in Chapter 4. Since, the system is pipelined at the same rate throughout, the network throughput is not affected significantly by the internode distance. This fact is seen in the simulations of the 384 node ShuffleNet and 400 node MSNet configurations. The plots show that while, the 2048 node ShuffleNet can support a mean time between accesses of 15 ticks under space-time configuration, the 2024 node MSNet requires a mean time

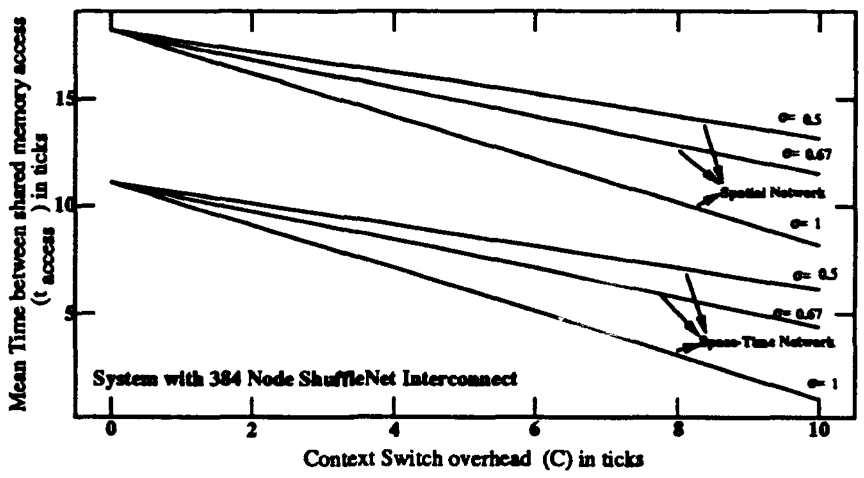


Figure 6.2. Mean time between shared memory access that is possible in a 384 node system with a ShuffleNet interconnect

between accesses of about 30 ticks even with space-time configuration.

The round trip latency determines the number of contexts required to keep the processor busy while it is waiting on shared memory requests from suspended contexts to be satisfied. If L is the round trip latency, then the minimum number of contexts required to keep the processor busy is given by

$$\begin{aligned} n_c &= \lceil \frac{L}{1/\sigma\eta_{max}} \rceil + 1, \\ &= \lceil \sigma\eta_{max}L \rceil + 1. \end{aligned} \tag{6.2}$$

Table (6.9) shows the value of n_c obtained for different system configurations using the simulation results for η_{max}, L and Eqn. (6.2). Since the networks tend to get rapidly unstable above link utilization of about 90%, the η_{max} values considered here as well as for the derivation of t_{access} are values at the knee of the latency throughput curves. Table (6.9) shows that the size and topology of the network does not have any significant effect on the number of threads needed to keep the processor busy. The reason for this behavior is the large flight latency component in the round trip latency as compared

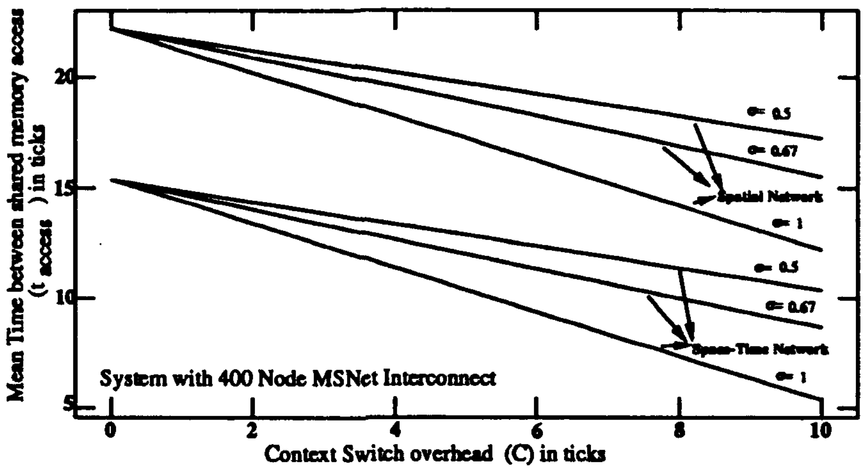


Figure 6.3. Mean time between shared memory access that is possible in a 400 node system with a MSNet interconnect

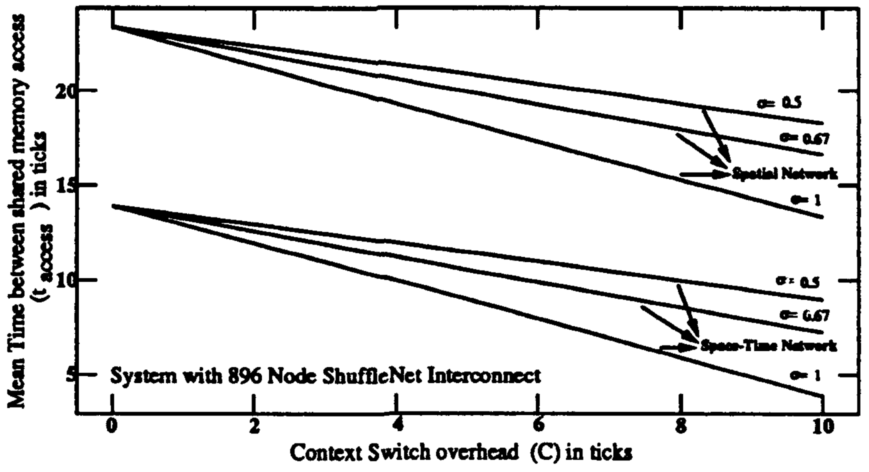


Figure 6.4. Mean time between shared memory access that is possible in a 896 node system with a ShuffleNet interconnect

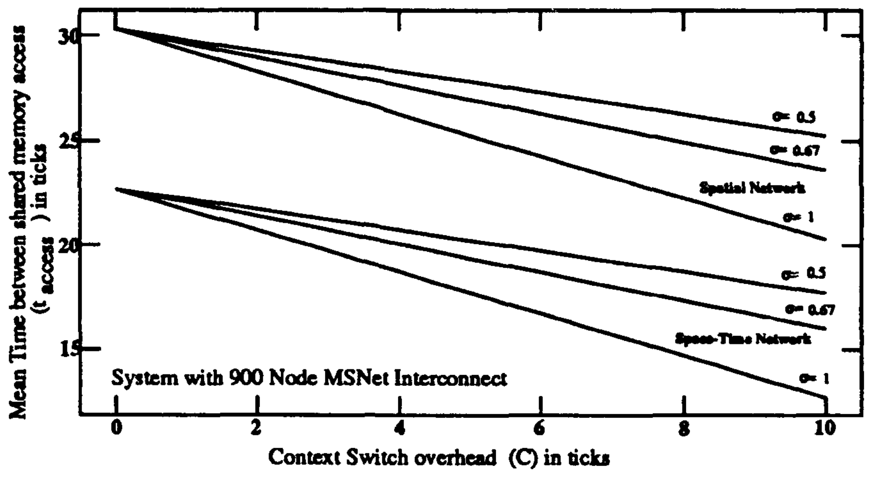


Figure 6.5. Mean time between shared memory access that is possible in a 900 node system with a MSNet interconnect

to other components put together. Since the system is pipelined, the round trip latency is approximately twice the product of $\langle E \rangle$, the average number of hops made by packets and n_l, the internode distance expressed in ticks. Under uniform traffic and steady state, there are as many requests as replies in the network at any given time. Hence, η_{max} $\langle E \rangle$ is the link utilization. Since, the knee of the latency throughput characteristics is reached around the same value of link utilization for all configurations, the number of contexts n_c is essentially proportional to n_l, the number of pipeline steps between two nodes. This is what is observed in the data shown in the Table (6.9). The simulations were carried out under the assumption that the spatial configuration had 10 pipeline stages and the equivalent number for the space-time configuration was 11 stages.

6.3 Behavior under Spatially Non-uniform Load

The preceding analysis was made under the assumption that processors generate requests uniformly distributed in space and time. In practice,

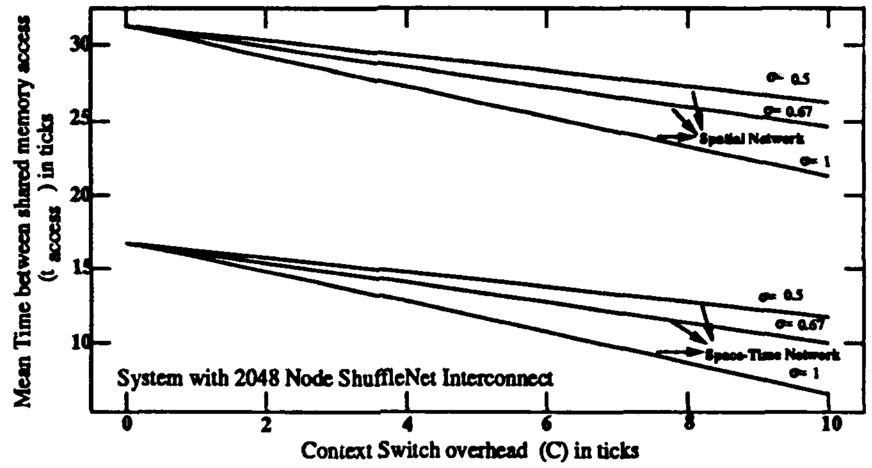


Figure 6.6. Expected value of mean time between shared memory access possible in a 2048 node system with a ShuffleNet interconnect

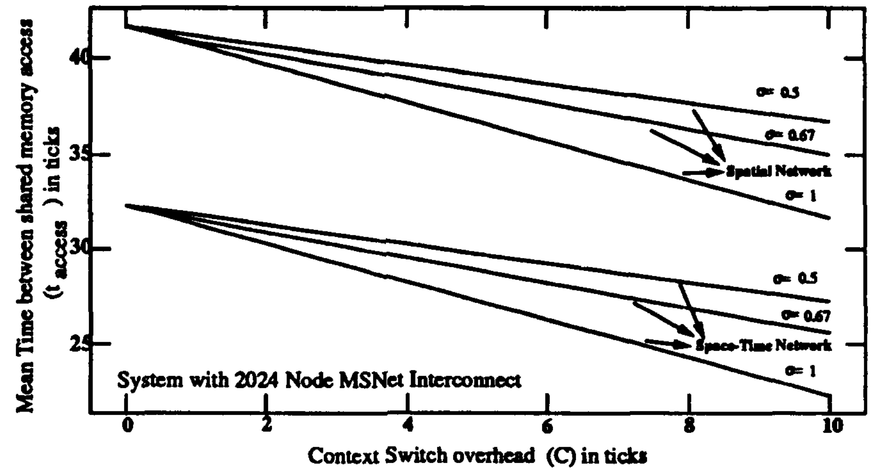


Figure 6.7. Expected value of mean time between shared memory access possible in a 2024 node system with a MSNet interconnect

Table 6.9. The minimum number of contexts required for various system configurations

Network size&type	# of Threads for σ=0.5	# of Threads for σ=0.67	# of Threads for σ=1.0
384SN(Spatial)	10	13	18
384SN(Space-Time)	11	14	21
400MS(Spatial)	10	12	18
400MS(Space-Time)	11	14	21
896SN(Spatial)	10	13	18
896SN(Space-Time)	11	15	21
900MS(Spatial)	10	13	18
900MS(Space-Time)	11	14	21

non-uniformities exist both in spatial distribution as well as in time. As mentioned in Section (6.1.2), our approach to study the behavior of the system under non-uniform workloads is to investigate the effects of spatial and temporal non-uniformities independent of each other. Spatial non-uniformities can take a variety of forms. The hot-spot phenomena causes a particular kind of spatial non-uniformity in access pattern due to generation of traffic that is targeted far more towards certain network resources than others. Such more often accessed resources, called hot-spots, are not a static feature of any system. Their numbers, locations and durations keep changing with time. The approach to this part of the study is similar to that used by Pfister and Norton [88] for buffered indirect multi-stage networks. Their single hot-spot model has been used by others to study the onset of hot-spot and subsequent recovery [90] as well as to study certain solutions to alleviate the problems caused by hot-spots [89] in such networks. The model has also been used to study the effect of hot-spots in packet-switched binary hypercube networks [91]. In this model, one particular memor module is designated as a hot module. A fraction ρ

of all shared memory references from a processor are directed towards the hot memory module and the remaining $(1-\rho)$ of their requests are distributed uniformly among all memory modules. We adopt a slight variation of the model, made to suit our organization of processors and memories. A memory module associated with one processor is chosen to be the hot memory module. We direct a fraction ρ of all other processor's shared memory references towards the hot memory module and the rest $(1-\rho)$ of their requests are distributed uniformly among all memory modules, except their own memory module. The processor associated with the hot memory module accesses memory modules at other nodes uniformly. The effective access rate of the hot memory module is $\eta(1-\rho)+\eta\rho(N-1)$. Since, the memory module can service only one request per tick, the above expression places a limit on the fraction ρ that accesses the hot memory module for any given average offered load η. The limit ρ_{max} is given by

$$\rho_{max} = \frac{1-\eta}{\eta(N-2)}. \tag{6.3}$$

with a corresponding link utilization of

$$\alpha = \frac{\langle E \rangle}{(1+\rho_{max}(N-2))}. \tag{6.4}$$

The four configurations studied in the last section, i.e. the 384 and 896 node ShuffleNets and the 400 and 900 node MSNets, were investigated under the hot-spot traffic outlined above. Two background loads equivalent to 50% and 80% link utilizations under uniform loading conditions were offered. For each configuration, simulations were carried out for different hot-spot probabilities ranging from zero to the value that turned the system non-steady. Figures (6.8) and (6.9) show the average round trip latency L as a function of the hot-spot probability ρ, obtained through simulations, for the 384 node

ShuffleNet and 400 node MSNet configurations. Similar results were obtained for 896 node ShuffleNet and 900 Node MSNet configurations. In all cases, the increase in round trip latency with increase in hot-spot probability is not significant until the system goes into the non-steady state. It is found that the throughput of the system is also not affected by the hot-spot traffic under steady state conditions. For a particular network configuration, a set of simulations were performed by fixing the average insertion bandwidth and varying the hot-spot probability from zero in steps of 1%. The system continued to stabilize without significant increase in the flight and round trip latencies of packets until a value of hot-spot probability very close to or greater than the maximum value suggested by Eqn. (6.3) was reached. At this value of hot-spot probability the system exhibited non-steady behavior with the round trip latency increasing sharply as seen at the end of all curves in Figs. (6.8) and (6.9). Equation (6.3) gives the allowable hot-spot probabilities of 2.7%, 4.0%, 4.5% and 5.8% for processor insertion bandwidths of 0.087, 0.062, 0.055 and 0.043 pkts/node/tick for the 384 node ShuffleNet configuration. The corresponding values for the 400 node MSNet system are 3.8%, 5.3% and 7.0% for processor insertion bandwidths of 0.062, 0.045 and 0.035 pkts/node/tick. The only wide discrepancy seems to be in the case of MSNet system with throughput 0.045 pkts/node/tick. The reason is that data points were collected at 5% and 6% probabilities. An intermediate value is expected to show non-steady behavior. A similar effect has been observed in buffered multi-stage interconnects [88, 89] as well as packet-switched hypercube network [91], except the degradation in network performance is far more in those cases.

An interesting fact to note is that the limit on hot memory access

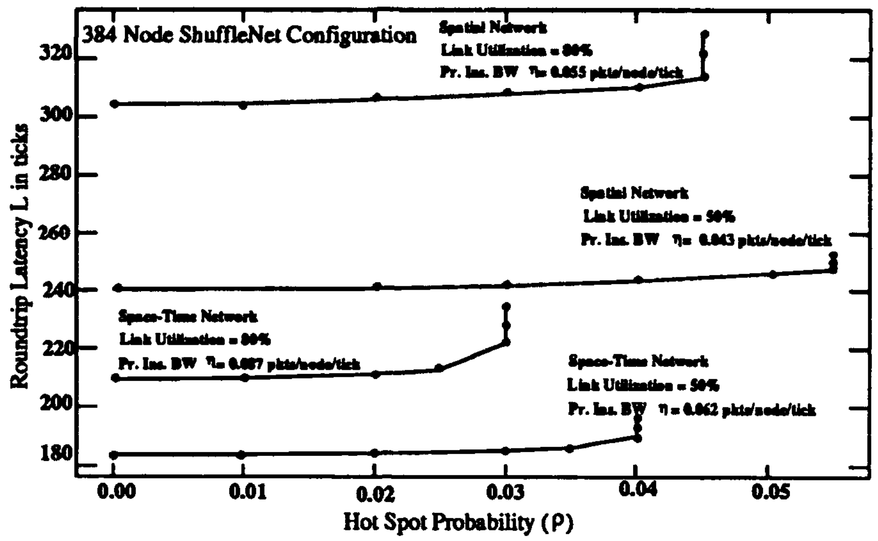


Figure 6.8. Roundtrip Latency as a function of hot-spot probability in the 384 node system with ShuffleNet interconnect

is not set by the network, but by the memory access bottleneck, which restricts the maximum access rate to one request per tick or cycle. Since a space-time network utilizes the network links more efficiently than an equivalent spatial network, it possesses a higher processor insertion bandwidth for the same link utilization. As a result the allowable hot-spot probability is less for the space-time configuration than for a purely spatial configuration used at the same link utilization. For a 256 node Omega network, the maximum possible bandwidth under uniform loading conditions was 0.6 pkts/processor/tick. 1% hot-spot traffic reduced the bandwidth to about 0.3 pkts/processor/tick, with 2% and 4% hot-spot ratios providing a net bandwidth of 0.18 and 0.10 pkts/processor/tick. The analysis of the buffered hypercube network presented in [91] assumes locality of reference, which is parameterized separately from the hot-spot probability. Their results are also qualitatively similar to the effect seen in Omega network. Figure (6.10) shows the variation in the round trip latency of packets in the 384 node ShuffleNet configuration as a function

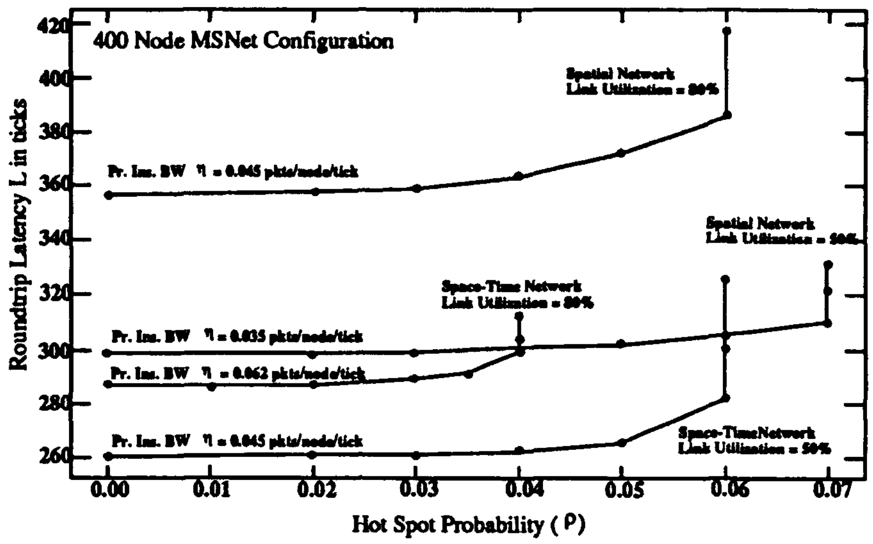


Figure 6.9. Roundtrip Latency as a function of hot-spot probability in the 400 node system with MSNet interconnect

of processor insertion bandwidth for various hot-spot probabilities. A significant difference between the buffered multistage interconnects and the ultrafast network is that the ultrafast network is able to tolerate a certain percentage of hot-spot traffic before the degradation starts and the rate of degradation, once it starts, is slower. There are a number of significant differences between the multi-stage interconnects and the ultrafast networks. Under normal operating conditions, the ultrafast networks provide a very high link bandwidth as compared to the requirements of the hosts, while the two are matched in the multi-stage networks. In the presence of hot-spots, bandwidth requirements at all other nodes of an ultrafast network are essentially the same as in the case of uniform load, except for much higher requirements placed on the hot modules. Even at those modules, the peak requirement is slightly above 1 pkt/node/tick as compared to a peak bandwidth of 2 pkts/node/tick that can be provided by the ultrafast network. In contrast, the normal requirement and the peak availability are the same in the case of multi-stage networks. Another significant

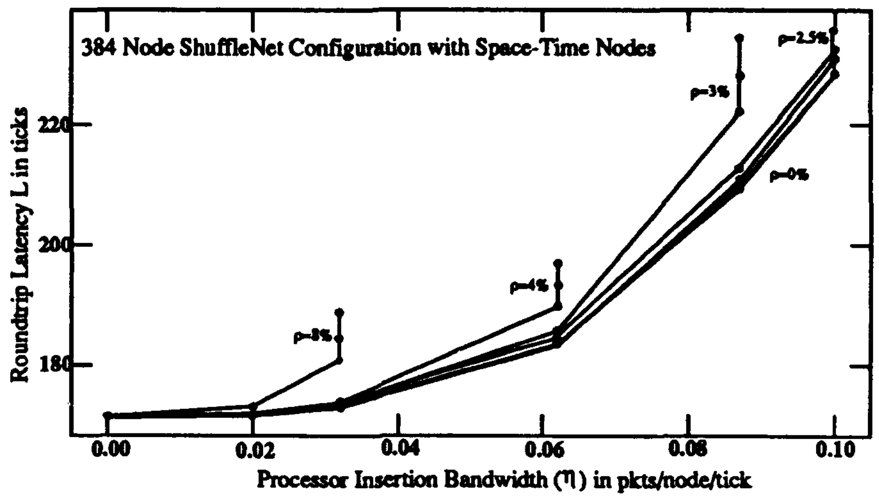


Figure 6.10. Latency - Throughput Characteristics of the 384 node system with space-time nodes for different hot-spot probabilities ρ

difference is that multi-stage networks provide a single path between source destination pairs, while in ultrafast networks multiple paths are available between source-destination pairs. In networks with *don't care* nodes, there are multiple minimum distance paths between many source-destination pairs. Deflection routing combined with these multiple paths tend to move the regular traffic as much as possible away from heavily utilized links. In multi-stage networks, links heavily utilized by the hot-spot traffic tend to block regular traffic, there by causing serious degradation in performance. A point to note is that, even under uniform loading conditions, the maximum usable insertion bandwidth is less than 1 for multi-stage interconnects. So, the continuous degradation in the bandwidth under hot-spot traffic in multi-stage networks is the combined effect of the network behavior and the memory bottleneck. In contrast, the response to hot-spots in ultrafast networks is only limited by the memory bottleneck, and is not affected further by the network.

A phenomenon analogous to the tree saturation in buffered networks, is the blocking of the insertion points at nodes that precede the hot-spot node in the ultrafast network. In our simulations, we allowed only one outstanding request for each entry port to the network. The processor was blocked from inserting any requests until it found a space in the interface unit buffer. The amount of time each processor was blocked was measured. Another alternative would have been to measure the packets in the buffer at the processor that feeds the network interface unit. Figures (6.11) through (6.13) show the the number of processor block-outs at each node during 1000 ticks for various hot-spot probabilities in a 384 node ShuffleNet system with Space-Time nodes. The processor insertion bandwidth was 0.087 pkts/node/tick. The hot-spot probability of 0.02 (2% hot-spot ratio) is within what the memory bandwidth allowed, while the other two cases were beyond the limit of the network. Hot-spot probability of 0.03 is just beyond 0.0275 allowed by the memory bandwidth. The effect of traffic non-uniformity can be seen in all the plots. In the case of 2% hot spot ratio, the average number of block outs is small compared to the round trip latency of the packets. The maximum number of block-outs is 45. With 3% hot-spot ratio, most nodes suffer less than 50 block-outs, with few nodes behind on the tree showing a block-out ranging up to 300. Note, that the network is non-steady at this hot-spot ratio. These results were collected after the simulation had been running for 4000 ticks. When the system is in steady state, it stabilizes within 1000 ticks. The number of processor block outs increased far more when the hot-spot ratio was increased to 4%, as seen in Fig. (6.13). In all other cases too, it was found that the number of processor block outs at different nodes was low until the system reached a non-steady state. The number of packets routed by nodes was also counted. Figures (6.14) through (6.16) show the number of packets routed by each node, during a 1000 tick interval, in the same system, for the same hot-spot ratios as above. The hot-node (node 0) routed the maximum number of packets as expected. While most nodes, routed 1550 to 1600 packets with hot-spot probability of 0.02, a few nodes on the tree behind routed significantly more. With a hot-spot ratio of 3%, the system exhibited a similar performance, with higher number of packets routed by all nodes, and with more variations than before. The hot-node routed the maximum of 2000 packets. Increasing the hot-spot ratio to 4% saturated the network almost uniformly with all nodes routing over 1900 to 2000 packets. In general, a drastic effect is seen near and beyond the maximum hot-spot probability that can be tolerated by the hot memory. At high link utilization and hot-spot probabilities slightly beyond the maximum steady state hot-spot traffic, the whole network filled up with packets. The processor block-out times became uniformly high and the nodes engaged in forwarding packets almost all the time.

6.4 Behavior under Temporal Bursts

This sections concerns the response of the model system to temporal variations in traffic. As in the case of spatial variations, temporal variations can, in general, take various forms. In order to separate the effects of spatial distribution, we create a synthetic workload that periodically turns all processors bursty at the same time. No new requests are generated by a processor until all packets of its latest burst enter the network. The burst sizes and burst interval are fixed during a particular simulation and are varied between different simulations. During every simulation, the system is initialized in a steady state under uniform load before turning the system into a burst mode. Apart

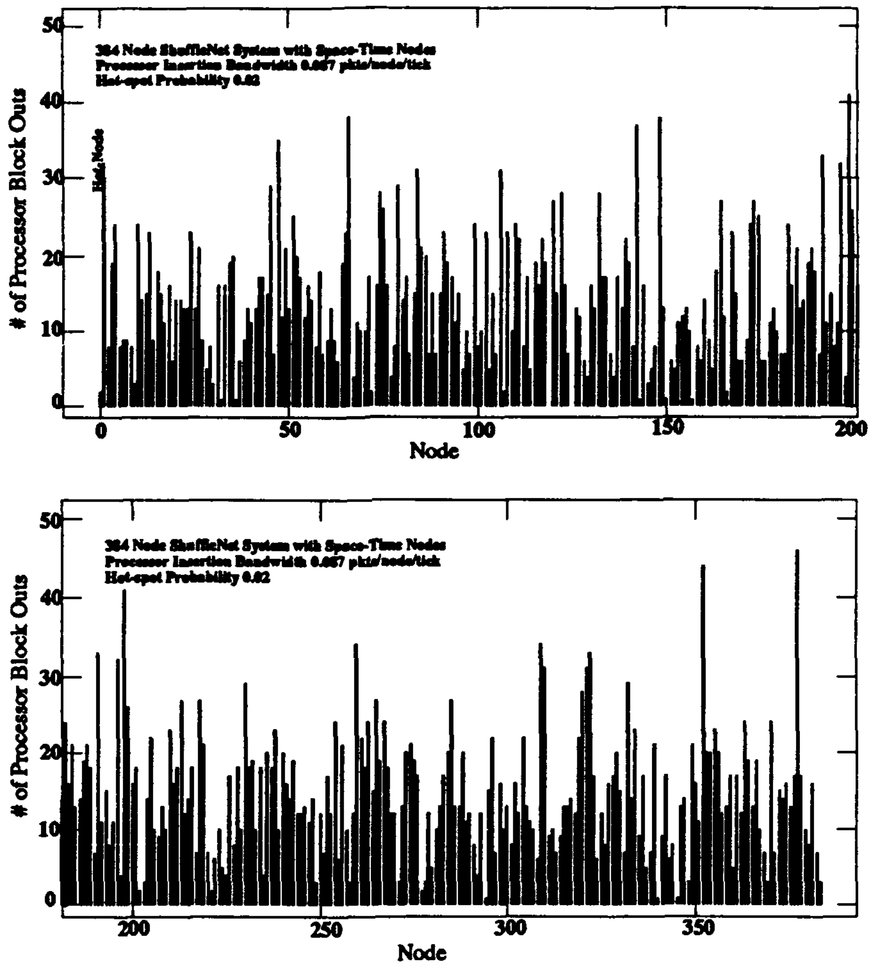


Figure 6.11. The number of processor block-outs at nodes of a 384 Node ShuffleNet System with space-time nodes, with 2% hot-spot ratio

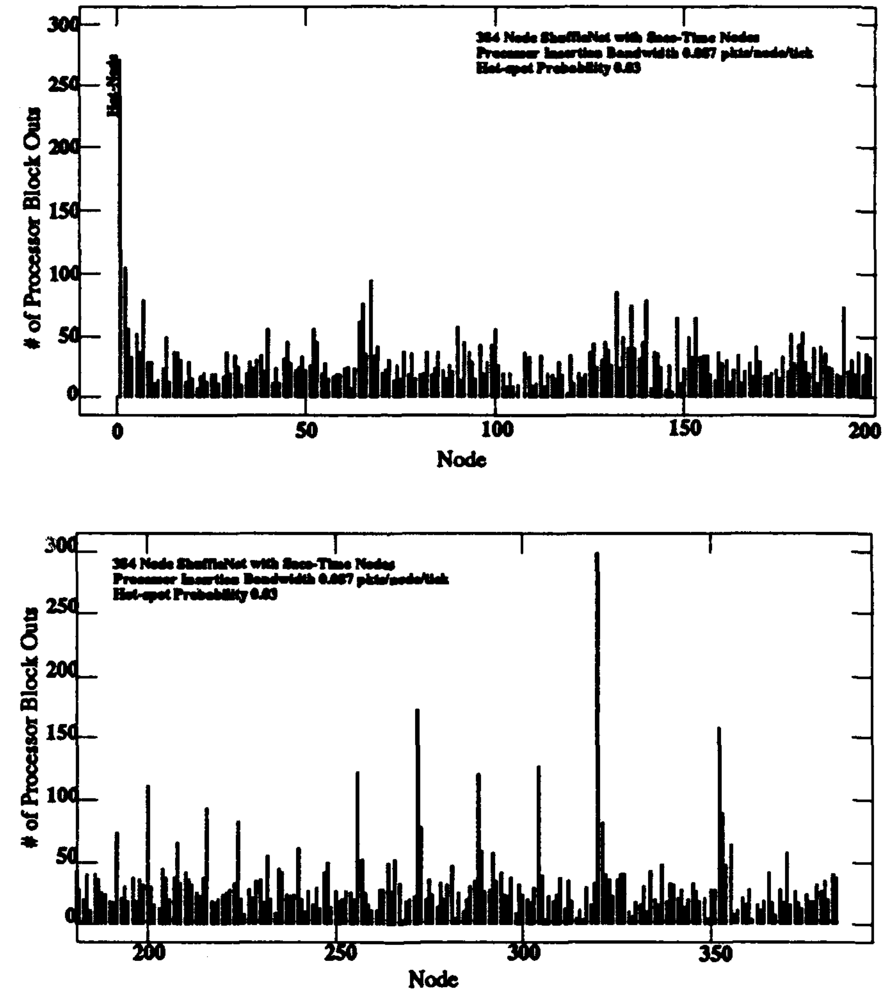
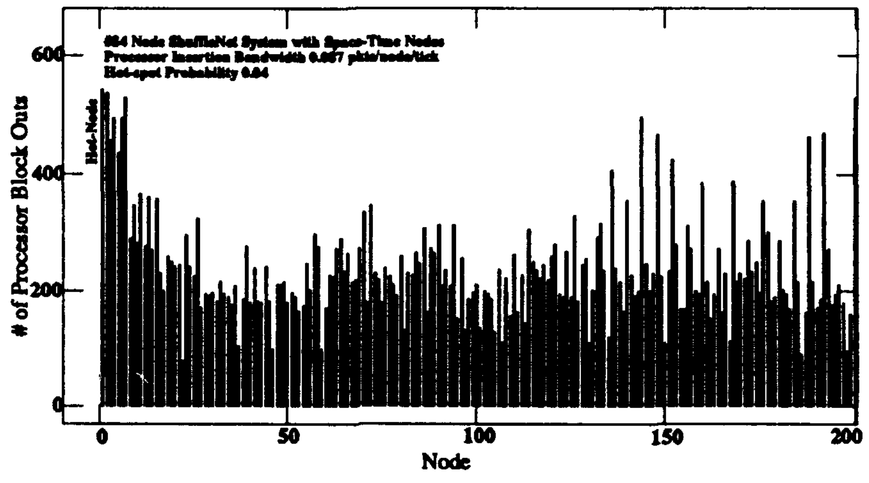


Figure 6.12. The number of processor block-outs at nodes of a 384 Node ShuffleNet System with space-time nodes, with 3% hot-spot ratio



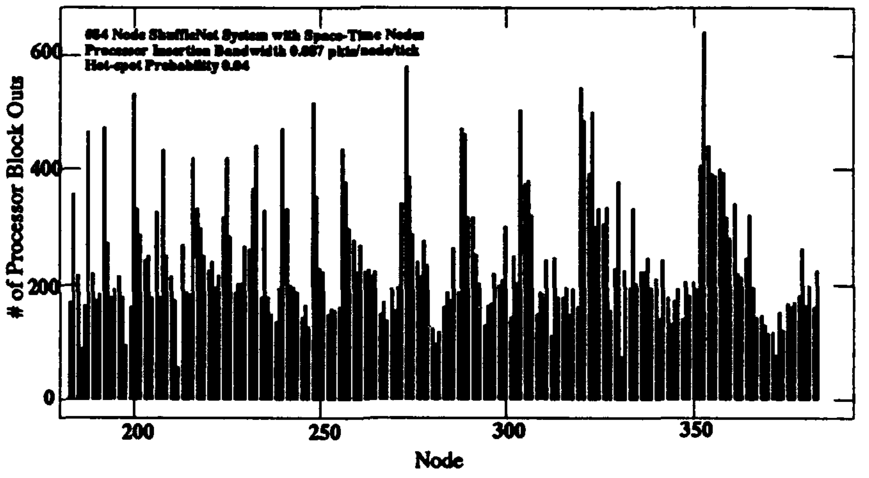


Figure 6.13. The number of processor block-outs at nodes of a 384 Node ShuffleNet System with space-time nodes, with 4% hot-spot ratio

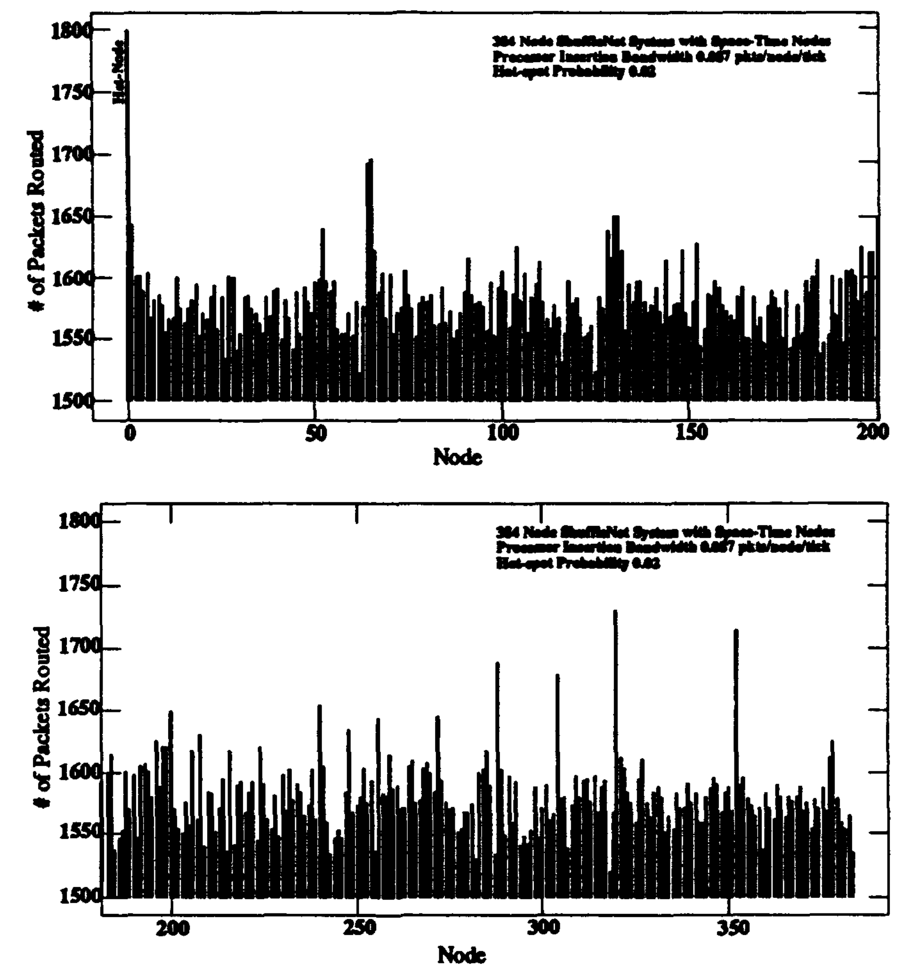


Figure 6.14. The number of packets routed by nodes of a 384 Node ShuffleNet System with space-time nodes, with 2% hot-spot ratio

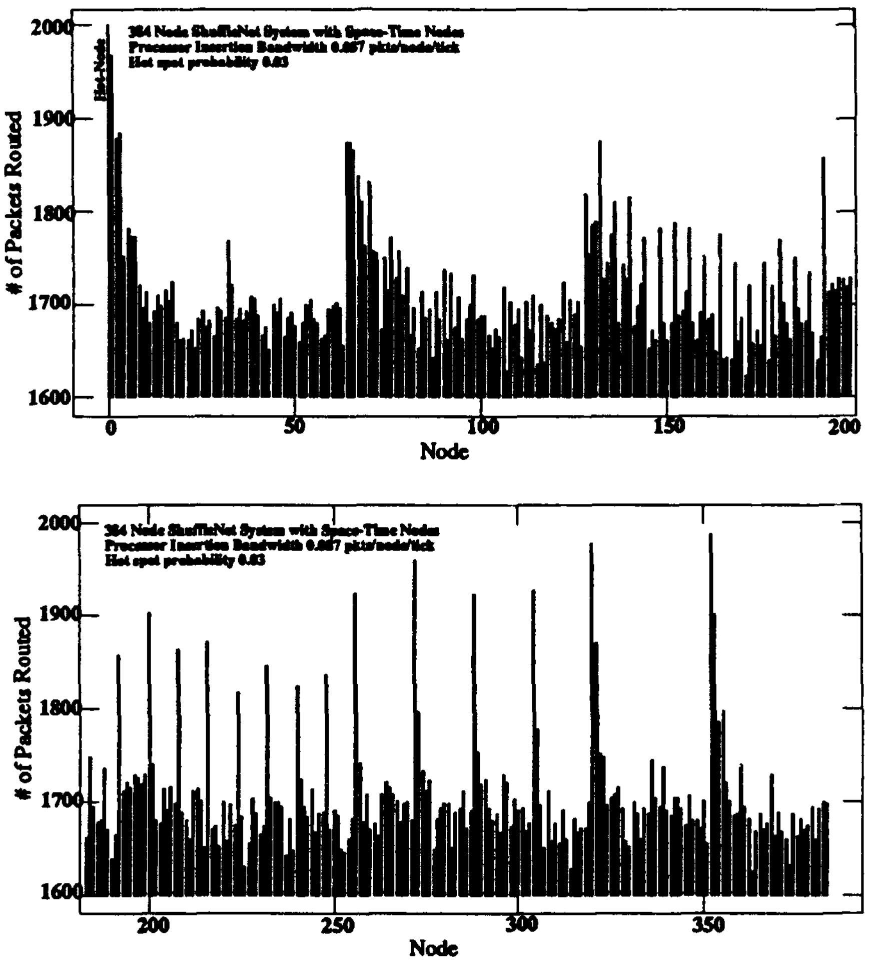


Figure 6.15. The number of packets routed by nodes of a 384 Node ShuffleNet System with space-time nodes, with 3% hot-spot ratio

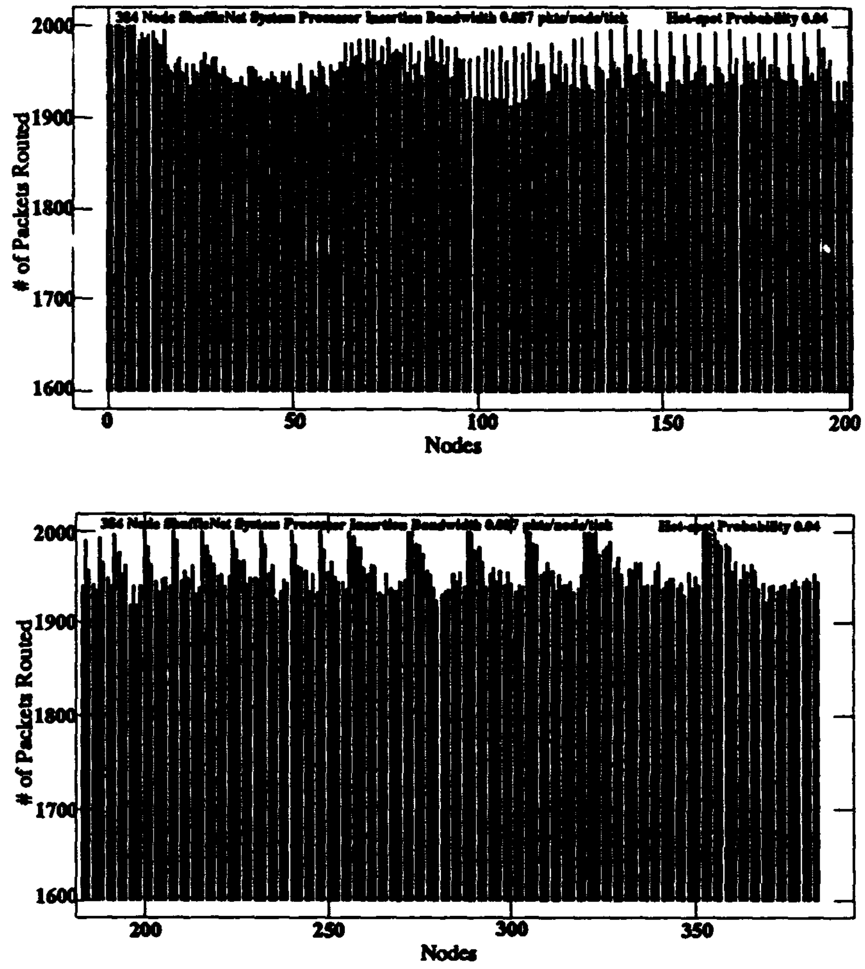


Figure 6.16. The number of packets routed by nodes of a 384 Node ShuffleNet System with space-time nodes, with 4% hot-spot ratio

from measuring the average throughput and latency of packets received between bursts, the relaxation behavior of the system following a burst was also studied. As soon as the processors turn bursty, they potentially want to inject one packet per tick until all the packets of their respective bursts are inserted into the system. Since, the network cannot accept packets at that rate, the processors are blocked out. The number of processor block-outs in the system during intervals of 10 ticks were measured. The first set of simulations were done by fixing η, the background insertion bandwidth of the processors and B, the burst size. The burst interval was varied. Figures (6.17) through (6.19) show the typical behavior of the system. These results were obtained for the 384 node ShuffleNet system with space-time nodes. The background processor insertion bandwidth, η was 0.087 pkts/node/tick, which is equivalent to 80% link utilization for that uniform load. The burst size B was 32 packets. The number of processor block-outs in the system was counted during each 10 tick interval. Immediately following the start of a burst, the number of processor block-outs rises sharply. Given enough time between bursts, the system ultimately relaxes to the steady state behavior under the uniform background load. Figure (6.17) shows the case when the burst interval T is just about enough for the system to fall back into the background steady state. If a new burst is started before the system reaches its background steady state, as in Figs. (6.18) and (6.19), the system still goes into a repetitive behavior. The only difference caused by the more frequent bursts is that, the system does not return to the uniform loading steady state behavior at any time. As the burst interval is decreased, the system ultimately gets overloaded and exhibits non-steady behavior. Thus, for any given set of values for the background

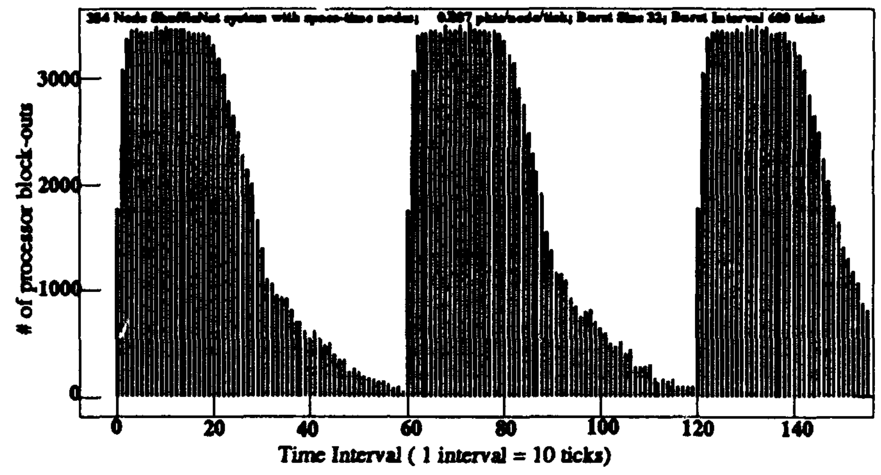


Figure 6.17. The number of processor block-outs in a 384 Node ShuffleNet System with space-time nodes, with burst size of 32 pkts and burst interval of 600 ticks (η = 0.087 pkts/node/tick)

insertion bandwidth η and the burst size B there is a limit on the minimum burst interval. As long as the burst interval is equal to or greater than this minimum, the network will be able to adapt to the bursts by settling into a steady periodic behavior.

The repetitive behavior suggests that over each burst interval the average processor insertion bandwidth remains the same. This was indeed observed during simulations, after the system settled into steady periodic behavior under the bursts. The average flight latency was however found to be greater than the average flight latency when the system is under a uniform load equal to the average processor insertion bandwidth under the bursts. For the 384 node ShuffleNet systems with burst interval of 500 ticks, the increase was within 10%. The reason for this increase is the increase in the total number of deflections due to higher probability of collision when the system is running at nearly full link utilization due to the bursts. Extensive parameterized simulations need to be done before the effect on latency can be quantified.

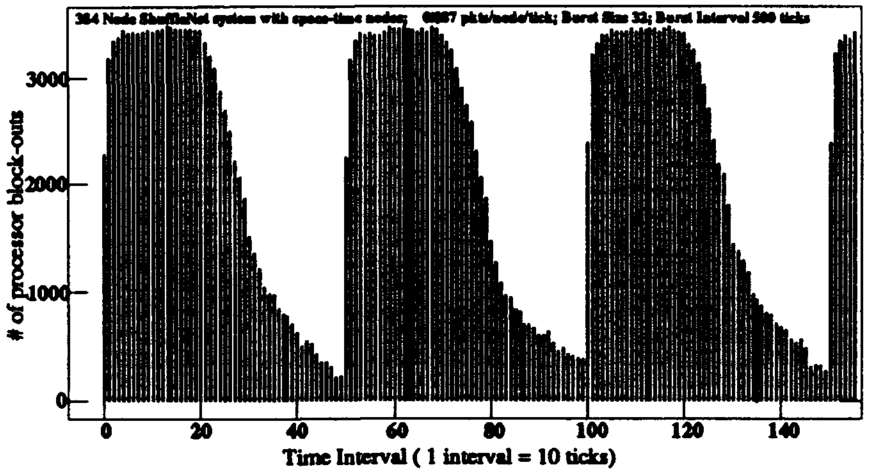


Figure 6.18. The number of processor block-outs in a 384 Node ShuffleNet System with space-time nodes, with burst size of 32 pkts and burst interval of 500 ticks (η = 0.087 pkts/node/tick)

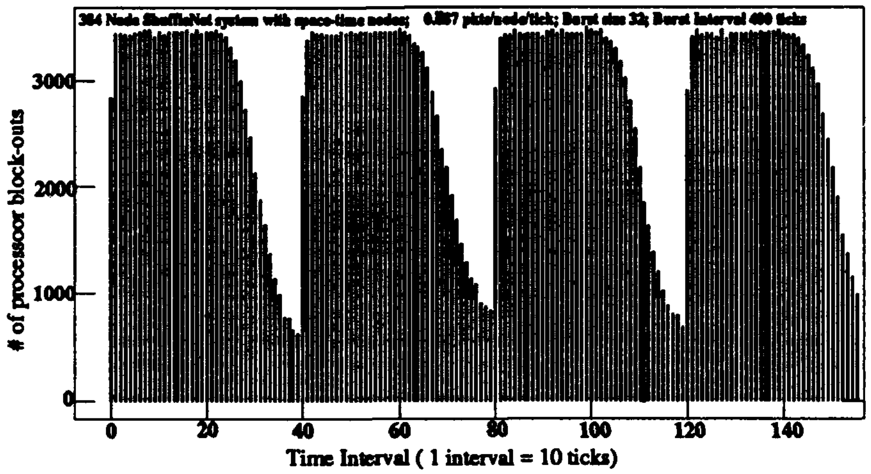


Figure 6.19. The number of processor block-outs in a 384 Node ShuffleNet System with space-time nodes, with burst size of 32 pkts and burst interval of 400 ticks (η = 0.087 pkts/node/tick)

Figures (6.20) and (6.21) show the effective or average processor insertion bandwidth, η_{eff}, measured during a burst interval of 500 ticks for the 384 node ShuffleNet systems for various burst sizes. The burst size was increased in steps of 4 starting from zero until the system showed a non-steady behavior. The last point where the curves stop, which is the same for all curves in the same plot, was the highest burst interval simulated before the system turned non-steady. The allowable maximum burst size for the specified burst interval is within this value and this value plus four. Whatever be the background uniform load, the system ultimately reaches a state in which the average insertion bandwidth is equal to the maximum processor insertion bandwidth under uniform load. A similar behavior is observed in the MSNet systems, as shown in Figs. (6.22) and (6.23). For MSNet with space-time nodes, a burst size of 30, between 28 and the next step 32, was also simulated because the curves did not get close enough at a burst size of 28 (Fig. (6.23)). For both topologies, we find that the space-time network adjusts better to bursts by allowing longer bursts within the same burst interval.

Another measure of the system behavior that can be derived from these observations is the time to inject a burst into the network, i.e. the time taken by the processors to return to the uniform injection state following the introduction of a burst. If the system takes T_b ticks to insert a burst of size B into the network, then the effective or average insertion bandwidth of the processors is given by

$$\eta_{eff} = \frac{(T - T_b)\eta + B}{T}, \tag{6.5}$$

where T is the burst interval. As seen from Figs. (6.20) through (6.23), the variation of η_{eff} with B is not linear, i.e. as the burst size is increased the network tends to saturate within a shorter burst interval. Equation (6.5) can be rewritten to express T_b as a function of η, B, T and η_{eff} as

$$T_b = \frac{B + (\eta - \eta_{eff})T}{\eta}. \tag{6.6}$$

Figure (6.24) shows T_b, computed using Eqn. (6.6), as a function of burst size for the 384 node ShuffleNet systems, systems without and with space-time nodes. One set of curves are for the system with uniform loads leading to 50% link utilization and the other for uniform loads that lead to 80% link utilization. Figure (6.25) shows similar curves for the 400 node MSNet systems. Both the ShuffleNet and the MSnet topologies exhibit the same behavior under the kind of bursty traffic offered. The time to inject a burst increases slowly with the burst size and becomes linearly dependent on the burst size for larger bursts. This is evident from Eqn. (6.6), in which the second term in the numerator tends to $(1 - \eta_{max}/\eta)T$ for larger values of B. The superiority of the space-time system over the equivalent spatial system is evident from the fact the bursts are injected more rapidly by the space-time system, thereby allowing larger bursts than an equivalent spatial system.

6.5 Summary of System Behavior

In this chapter we have presented a simulation based analysis of a model multiprocessor system interconnected by an ultrafast network. The final model was abstracted in steps from a more general formulation of the system. Initially a number of assumptions were made to bind the general parameter space of the system to an intermediate model system. While these assumptions concerned system parameters and the system characteristics, a

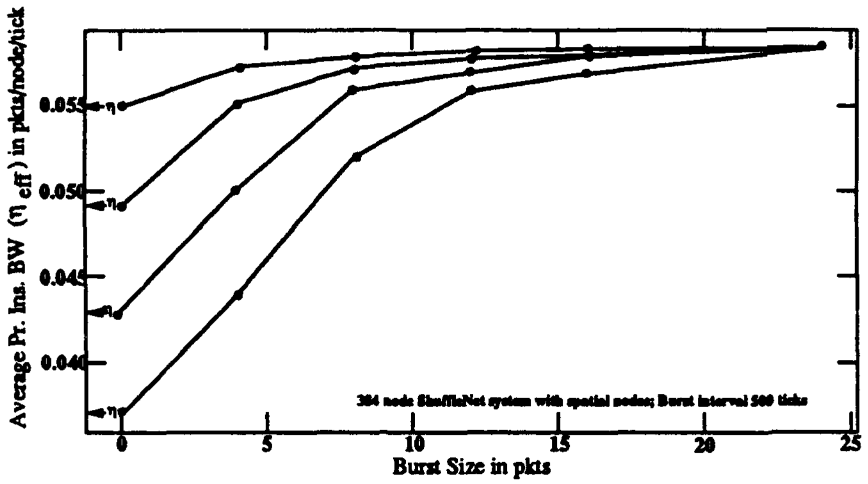


Figure 6.20. The effective or average processor insertion bandwidth η_{eff} as a function of burst size in a 384 node spatial ShuffleNet system with burst interval of 500 ticks for different uniform background loads

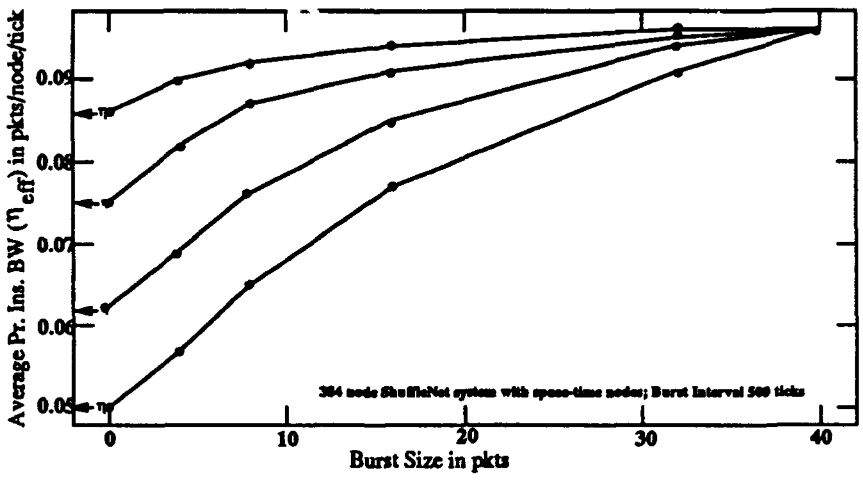


Figure 6.21. The effective or average processor insertion bandwidth η_{eff} as a function of burst size in a 384 node space-time ShuffleNet system with burst interval of 500 ticks for different uniform background loads

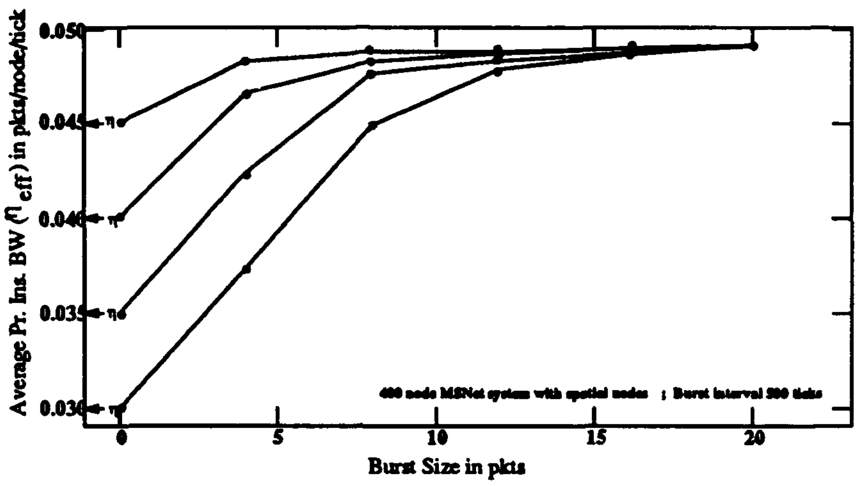


Figure 6.22. The effective or average processor insertion bandwidth η_{eff} as a function of burst size in a 400 node spatial MSNet system with burst interval of 500 ticks for different uniform background loads

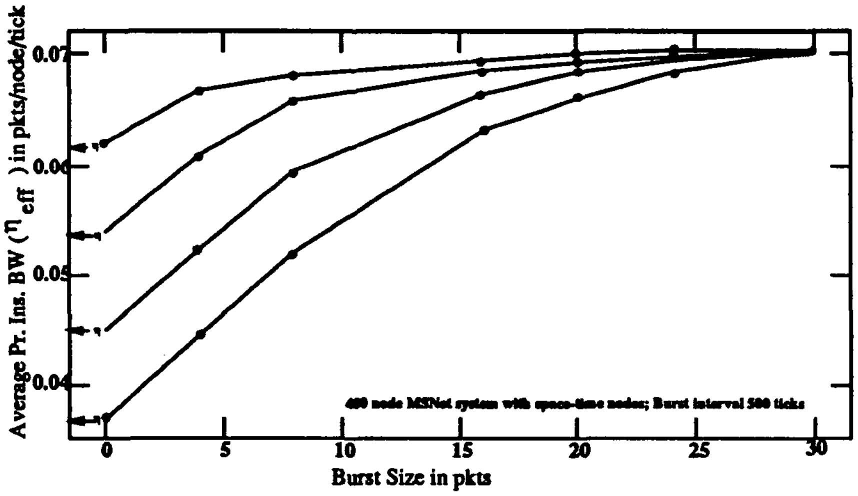


Figure 6.23. The effective or average processor insertion bandwidth η_{eff} as a function of burst size in a 400 node space-time MSNet system with burst interval of 500 ticks for different uniform background loads

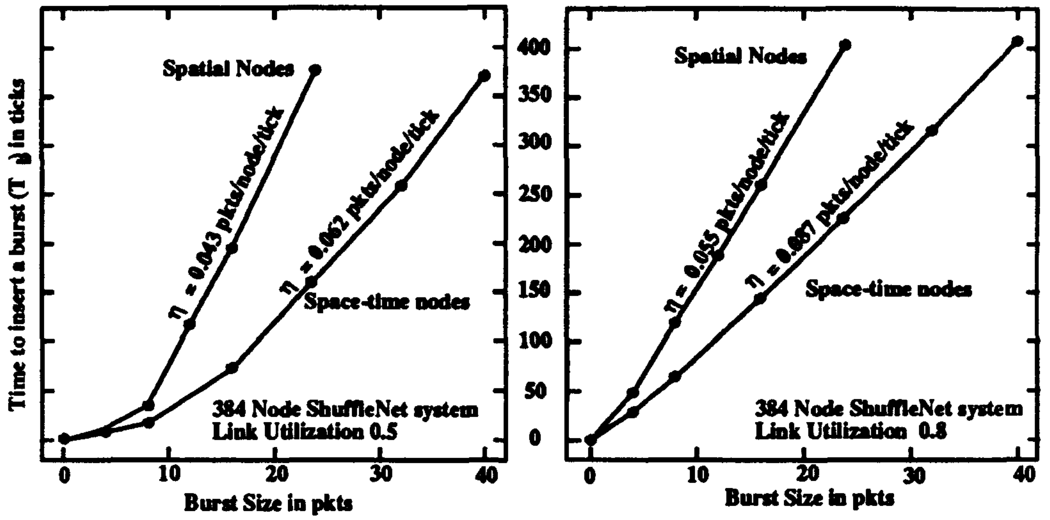


Figure 6.24. The time to insert a burst into the 384 node ShuffleNet system, with burst interval of 500 ticks for two different link utilizations under uniform background loads

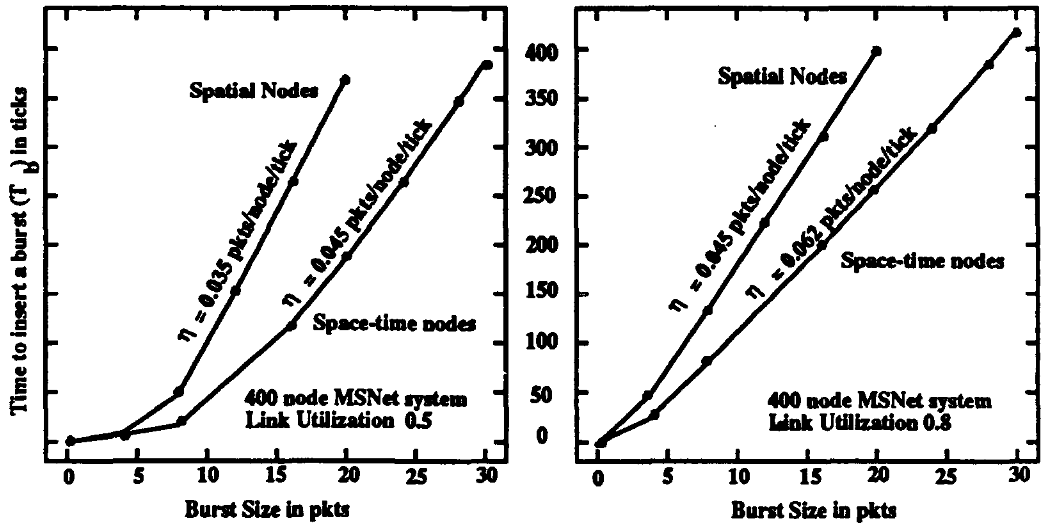


Figure 6.25. The time to insert a burst into the 400 node MSNet system, with burst interval of 500 ticks for two different link utilizations under uniform background loads

more abstracted model was created by simplifying the functions of the processors and the memories. The workloads presented were formulated to study the behavior of the system under some specific traffic conditions. Three different traffic patterns were studied. While the first was an uniform traffic, the other two were non-uniform patterns created by purely spatial and purely temporal variations in request patterns. The standard hot-spot model that has been traditionally used to study non-uniform references in address space was used. Behavior under temporal variations was studied by avoiding spatial non-uniformities.

The study of the behavior under uniform load provided a basic idea about the behavior of ultrafast networks in multiprocessor system. The topology of the system plays an important role in determining the available processor insertion bandwidth and the latency of messages. Space-time switching was found to be an efficient way to improve the performance of deflection routed interconnects. The number of contexts or threads, needed to keep the processors busy while waiting for memory access requests to be satisfied, was found to be dependent only on the internode link length and the fraction of memory access requests on which a thread waits for a reply. The size and the topology had no effect on the number of threads needed. Space-time switching affects the number only through the increase in internode distance due to extra pipeline stages at switching nodes.

The system adjusted quite well to the kind of temporal variations that were presented. The behavior is predictable as long as the average processor insertion bandwidth between bursts is within the maximum bandwidth available to the processors. Bursts being a temporal phenomenon, the space-time network performed better under temporal bursts. They allowed bigger bursts for the same burst interval and at the same background link utilization. The result is far more significant because of the fact that, for the same background link utilization, the space-time network provides a higher processor insertion bandwidth than a purely spatial network.

The behavior in the presence of hot-spots is influenced by the processor insertion bandwidth and the memory bottleneck which restricts maximum memory access bandwidth to one. Hot-spots being a purely spatial phenomenon, space-time switching does not influence the response of the system to the presence of hot-spots. A comparison of the results with the performance of multi-stage store-and-forward networks obtained by others shows that deflection routed direct networks can tolerate hot-spots far more than multi-stage store-and-forward networks. Table (6.10) provides a comparison between two 2048 processor systems, one with ShuffleNet topology and the other with Omega network [92] topology. An Omega network is a logarithmic multi-stage network with perfect shuffle interconnection between nodes of adjacent stages. While the results on the ShuffleNet are based on our simulations, the performance measures given for the Omega network are optimistic extrapolations from the results on a 256 processor system presented in [89].

Though a bandwidth of 1 pkt/processor/tick is theoretically possible in the Omega Network configuration, it is not possible to achieve this bandwidth in systems with finite buffers because of contentions in the network [93]. With store-and-forward buffers of size 4 at each output to a node, the 256 node Omega network system has been shown to be limited to about 0.6 pkts/processor/tick under uniform traffic conditions [89]. The bandwidth

Table 6.10. A comparison of a 2048 processor ShuffleNet system with a 2048 processor Omega Network system

	ShuffleNet	Omega Network
number of processors and memories	2048 processor-memory 2048 pairs attached to 2048 nodes	2048 processors on one side and memories on the other
ports per processor or memory	2 ports per processor-memory pair	1 port per processor or memory
number of nodes	8×2^8 = 2048 nodes	$1024 \times log_2 2048$ = 11264 nodes
switches per node	Three 2×2 switches one with full and two with simple function	Two 2×2 switches both with full function
routing paradigm	deflection with restricted dynamic buffering using space-time permuters	store-and-forward with static buffers
Average network latency	ideal: 21 hops real: 26 hops	ideal: 20 hops ??
number of threads	14	90 - 120 (estimate)
maximum processor insertion bandwidth	0.06 pkts/node/tick	0.6 pkts/node/tick (optimistic estimate)
maximum hot-spot probability	0.8% results in 80% link utilization	0.03% results in 60% link utilization
processor insertion bandwidth of 0.06 pkts/node/tick	same as above	expected hot-spot probability is 0.8% with 6% link utilization

of a 2048 processor Omega network system will be lower than this limiting value for 256 processor system. This study shows that a 2048 node ShuffleNet system can support a maximum processor insertion bandwidth of 0.06 pkts/node/tick. A major difference between the two configurations can be seen in their response to hot-spots. While the direct ShuffleNet configuration does not show any degradation in performance until the hot-spot probability reaches 0.8%, the indirect Omega network configuration starts to degrade continuously as the hot-spot probability is increased from 0% [89]. The main reason is that the ShuffleNet provides a peak memory access bandwidth of 2, while only half that can be utilized because of the memory bottleneck. Also, the deflections cause regular traffic to move away from congested links. The Omega network system can support only about 0.03% hot-spot traffic at the projected optimistic maximum offered load of 0.6 pkts/processor/tick. If the system is offered only 0.06% pkts/processor/tick, then the Omega network system is also expected to support a hot-spot probability of 0.8%. The resulting link utilization will be just about 6%. Thus, with nearly 1/5th the number of nodes, the ShufflNet configuration should be able to provide about the same performance as the Omega network configuration for real loads which tend to be highly non-uniform both in space and time.

CHAPTER 7

SUMMARY, CONCLUSIONS AND FUTURE WORK

7.1 Summary

An ultrafast network provides a very high link bandwidth between nodes as compared to the insertion bandwidth sought by the hosts. Most direct networks provide an access bandwidth which is much smaller than the link bandwidth. Also, the bandwidth differential in such systems increases with the number of nodes in the system. In contrast, indirect networks can provide an access bandwidth comparable to the network link bandwidth. So, though it is possible to turn an indirect network into an ultrafast network by designing the system such that the processors' need to access the network is far less than what the network can provide, it is not a restriction placed on the system as in the case of direct networks.

Another major difference between direct and indirect networks is that while direct networks can operate both under store-and-forward and deflection routing schemes, indirect networks are traditionally operated only under the store-and-forward scheme. There are a number of trade-offs between the two routing schemes. One of the principal advantages of deflection routing that makes it more attractive than store-and-forward routing is its suitability for efficient optoelectronic network implementations. The motivation for this study was provided by the need to exploit the potentials of such implementations, while mitigating their weaknesses.

Interconnection networks for shared memory multiprocessors generally provide a network link bandwidth to processors that is comparable to their insertion bandwidth requirement. So, the main purpose behind the investigations reported here was to characterize the impact of a class of ultrafast networks, direct networks that operated under a deflection routing protocol, on the performance of shared memory multiprocessors. The specific goals set at the start of this work were:

- To characterize the performance of ultrafast networks under deflection routing using one way messages.
- To study the impact of topological differences on the performance of the network
- To identify the basic system requirements that are needed to use ultrafast networks in shared memory multiprocessors.
- To differentiate the effect of the network on the processor insertion bandwidth from its effect on message latency.
- To characterize the performance of the system under well defined variations in network traffic caused by variations in offered load.
- To investigate the scalability of the system in number of nodes, and in internode distance.

While the first two goals listed above concern the raw network performance, the others are related to the performance of the multiprocessor system as a whole. Hence, the work done towards this study was divided into two parts that addressed the raw network performance and the composite system performance.

The primary contributions of this research effort, in the order in which they resulted, are:

- Built an object oriented network simulator in C++, a generic tool that separates the hosts and the network though a network interface unit, and allows independent variations in network configuration, routing protocol and the processor-memory configuration.
- Analyzed the behavior of the logarithmic SuffleNet using its unique properties and obtained a simple closed form model to predict its behavior.
- Identified the issues that complicate a similar analysis of the Manhattan Street network (a two dimensional toroidal network).
- Identified the limitations of deflection routing, through the above mentioned theoretical analysis as well as extensive simulations.
- Proposed the idea of limited space-time switching at the nodes of a fast packet-switched network to partially overcome the negative effects of deflection routing, and thus enhance the performance of the network.
- Predicted the potentials of an implementable scheme for space-time switching in fast packet-switched networks, through a topology independent theoretical analysis. The analysis showed that, using limited temporal reordering of packets, the network can achieve a performance that is close to a collision free ideal network. The predictions were verified through simulations for both topologies mentioned above.
- The analysis of the specific space-time switching technique proposed in this work and some recent, independent work by others on similar techniques that involve switching in space and time led to a generic classification of the possible variations under the general space-time switching paradigm.

- The study of the model system under uniform load led to the identification of the role played by ultrafast networks in supporting the requirements of the processor-memory subsystem.
- The study led to the identification of behavioral differences between deflection routed direct networks and multi-stage indirect networks in multiprocessor systems with hot-spots.
- The study of the effect of temporal bursts in the system demonstrated the capability of ultrafast networks to adapt to temporal variations as long as they are within the limit that would saturate the entire network.

7.2 Conclusions

The first part of the study concerned the performance of the network under deflection routing. The key to improving the performance of deflection routing networks is to reduce the probability of deflection through reducing collisions in the network. A factor that influences the probability of deflection is the number of *care* nodes encountered by packets during their flight through the network. The lower the number *care* nodes encountered by packets, the lower will be the probability of deflection. The network topology should be such that the network has an acceptably low average internode distance, a small penalty for deflection and sufficiently low fraction of *care* nodes, so that the performance of the network is close to that of an equivalent collision free ideal network. The analysis of the behavior of the two widely different topologies, i.e. the logarithmic ShuffleNet and the two dimensional toroidal Manhattan Street network, led us to believe that it will be a hard problem to identify a network topology that satisfies this condition.

It was identified that if packets could be reordered in time at the nodes, it would be possible to reduce the probability of deflection. A detailed theoretical analysis validated by simulations shows that, under uniform loading conditions, the network nodes with the kind of limited space-time switching that is proposed exhibit a latency behavior that is close to the ideal collision free network. We have focussed our attention throughout on nodes with two input links from and two output links to other nodes in the network. While considering temporal reordering we consider a sliding window of two time-slots, so that during each network cycle two consecutive slots on the two spatial channels are considered for routing the packets that may be present in them.

In direct networks, the insertion bandwidth available to the sources is dependent upon the latency of packets. Throughout this study we have assumed the network to be pipelined, with all internode links of the same length. In such pipelined systems, the available bandwidth is inversely proportional to the average number of *hops* made by the packets. Thus, space-time switching at nodes provides the network with higher average packet insertion rate as compared to the equivalent network with purely spatial switching at nodes.

Another important conclusion from the first part of the study on the network behavior is that the logarithmic ShuffleNet becomes far more efficient than the Manhattan Street network (MSNet) as the network size is increased. ShuffleNets provide smaller latency and higher throughput than equivalent MSNets. Though it may appear that this result is obvious because of the relative magnitudes of the average internode distances in the two networks, it is not so straightforward because of the higher penalty for deflection in the case of ShuffleNets of depth greater than 4 as compared to MSNets. For large networks, of more than 2000 nodes, the relative degradation in performance of a ShuffleNet from its own ideal behavior is much higher than the similar degradation in the performance of an equivalent MSNet from its own ideal behavior. But, the negative effects are more than offset by the advantage provided by the logarithmic internode distance in the ShuffleNet to provide a overall better performance.

We have used a highly abstracted model of a fully pipelined shared memory multiprocessor system in this study. A number of simplifying assumptions have been made, partly to extract the fundamental behavior of the system and partly to reduce the complexity of the simulations. The basic conclusion from the study of the behavior of the model system under uniform workloads is that it will be possible to adjust the design space of the system to exploit the potentials offered by the ultrafast regime, while avoiding compromise on processor utilization. As was observed in the raw network behavior, ShuffleNet configuration with space-time switching provides optimal performance among the four configurations investigated.

The study showed that multithreading is important to keep the processor utilization up, and the number of threads needed is small for a closely spaced system. An important finding on this aspect is that the number of threads needed to keep the processor busy is dependent only on the internode link length and the fraction of memory accesses on which the processor switches context. The number of threads needed was approximately twice the number of packet slots between the input of one node to the input of the next when the processor context switched on every memory access. Space-time switching will affect the number only though an increase of one or at most two pipeline stages in the internode distance, needed to perform temporal reordering.

The study of the hot-spot effect led to the conclusion that the *memory bottleneck* is the limiting factor on the performance of the network as well as the system under hot-spots. The behavior of the system, both latency as well as bandwidth did not change significantly as the fraction of hot-spot traffic was raised from zero in an uniformly loaded network to a value that led to a required memory access bandwidth of one from the hot memory module. The behavior under hot-spots was not further limited by the network. The results demonstrate the adaptive nature of deflection routing that moved the regular traffic away from congested links. These characteristics of a deflection routed direct network are in contrast to the behavior of indirect networks. In indirect networks, the performance starts to degrade steadily as soon as the hot-spot ratio is increased from zero. Due to the rapid degradation in performance under hot-spots indirect networks provide much smaller memory access bandwidth to the processors than what is expected of them. This study shows that direct networks with far fewer switch nodes may be able to provide a performance comparable to that of an equivalent indirectly connected system.

A very simple analysis of the system behavior was done under temporal bursts. The system adjusted well to the kind temporal variations that were presented, as long as the average insertion bandwidth required by the sources did not exceed what the network can provide. The average latency under bursty conditions is found to be slightly higher than the corresponding value for a unformly loaded system with the same average processor insertion rate. The limited data collected did not allow formal quantification of the effect. As expected, the space-time configurations showed greater adaptability to temporal variations in network access pattern.

7.3 Future Work

While the current study has provided a fundamental insight into the behavior that can be expected from ultrafast networks in multiprocessors, the work can be extended in a number of directions. In fact, there is a need to do so in order to design an efficient, predictable system using ultrafast interconnects. Some of the open areas for research are:

- Introduction of a real processor-memory model into the simulator.
- Formulation of more realistic workloads using simple high level shared memory constructs such as barriers, critical sections, etc.
- At the next level of investigation, comparison needs to be made between implementation choices of the constructs studied.
- A different issue that can be addressed is the effect of varying the network cycle time with respect to the processor-memory cycle time.
- The present simulation study, as well as other studies with some or all of the above requirements or possibilities, needs to be addressed for an equivalent direct network with store-and-forward routing instead of deflection routing.

Future network architectural studies will be devoted to the enhancement of the current design of Boulder Ultrafast Fiber-Optic network, and to provide guidance when implementation compromises are forced on us. It may also be that we don't yet know the optimum topology, that the off-the-shelf user processor systems could be slightly altered to greatly increase aggregate system performance, that retrofitting this interconnect into a current computer center that initially seems to be an 'obvious' improvement won't be, or that many other might situations arise that require considerable analytical and simulation resources to treat adequately. We thus feel that maintaining our architectural analysis and simulation capabilities is essential. A continued effort in this direction is particularly important to address the issue of scalability of the system. More often than not, solutions to problems on a small scale do not perform well when applied to medium- or large-scale situations. In the present context, we are focusing on scalability both from the view of large number of nodes as well as the increase in internode distance by orders of magnitude. Initial studies have shown that space-time switching nodes scale very well, much above the capability of purely spatial networks.

A full parametric simulation study of the source to destination capacity of the network under variations in topology and routing control algorithm is needed to direct the design of the deflection routing controller. The implementation of the high speed, pipelined controller will also influence which parameters to vary and their appropriate ranges. The parametric study and controller design will thus proceed in parallel. Initial studies have shown that limited use of time slot interchange (TSI) in packet switched networks can reduce contention significantly. TSI architectures developed here exploit the ease of providing precise delays in optical transmission. Controller design and network simulation will determine the most effective way of using this advantage. The architectural analysis and simulation studies will proceed in parallel with hardware development with a view to continually assess the architecture in situations not amenable to direct experimentation and measurement.

BIBLIOGRAPHY

[1] A. V. Ramanan, **Ultrafast Space-Time Networks for Multiprocessors**. PhD thesis, University of Colorado at Boulder, Department of Electrical and Computer Engineering, 1993.

[2] A. V. Ramanan, H. F. Jordan, and J. R. Sauer, "Space-Time Switching in Fiber-Optic Packet Switched Networks," in **Proceedings of the SPIE, Conference on Multigigabit Fiber Communication Systems**, Vol. 2024, (San Diego, CA), July 1993.

[3] A. V. Ramanan, H. F. Jordan, and J. R. Sauer, "Space-Time Networks for Optical Communications," in **Proceedings Thirty-First Annual Allerton Conference on Communication, Control, and Computing**, (Sponsored by the University of Illinois, Urbana-Champaign, IL), Sep 29- Oct 1 1993, pp. 566–575.

[4] A. V. Ramanan, H. F. Jordan, J. R. Sauer, and D. J. Blumenthal, "An Extended Fiber-Optic backplane for Multiprocessors," in **Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences**, Vol. 1, (Maui, Hawaii), Jan. 1994, pp. 462–470.

[5] A. V. Ramanan, D. J. Blumenthal, H. F. Jordan, and J. R. Sauer, "A High Speed Fiber-Optic Network for Short Packets," **Journal of high Speed Networks**, submitted for publication.

[6] M. D. Sprenger, "Trace-driven Simulation of multithreaded RISC Processors Connected via a Packet-switched Optical Interconnection Network Using Deflection routing," OCS Technical Report No.93-17, OCS Center, University of Colorado, Boulder, CO 80309-0525, 1993.

[7] P. Baran, "On Distributed Communication Networks," **IEEE Transactions on Communication Systems**, vol. 12, pp. 1–9, 1964.

[8] J. S. Kowalik, ed., **Parallel MIMD Computation: HEP Supercomputer and Its Applications**. Cambridge, MA: MIT Press, 1985.

[9] R.Alverson, D.Callahan, D.Cummings, B.Koblenz, A.Porterfield, and B.Smith, "The Tera Computer System," in **International Conference**

on Supercomputing, June 1990, pp. 1–6.

[10] A. S. Acampora, M. J. Karol, and M. G. Hluchyj, "Terabit Lightwave Networks: The Multihop Approach," **AT&T Technical Journal**, vol. 67, no. 67, pp. 21–34, 1987.

[11] A. Krishna and B. Hajek, "Performance of shuffle-like switching networks with deflection," in **Proceedings of IEEE INFOCOM '90, The Conference on Computer Communications**, IEEE Computer Society Press, 1990, pp. 473–480.

[12] N. F. Maxemchuk, "Regular mesh topologies in local and metropolitan area networks," **AT&T Technical Journal**, vol. 65, pp. 1659–1685, September 1985.

[13] A. G. Greenberg and B. Hajek, "Deflection Routing in Hypercube Networks," **IEEE Transactions on Communications**, vol. 40, pp. 1070–1081, June 1992.

[14] T. H. Szymanski, "An analysis of 'hot-potato' routing in a fiber optic packet switched hypercube," in **Proceedings of IEEE INFOCOM '90, The Conference on Computer Communications**, IEEE Computer Society Press, 1990, pp. 918–925.

[15] S. V. Ramanan, H. F. Jordan, and J. R. Sauer, "A New Time Domain, Multi-Stage Permutation Algorithm," **IEEE Transactions on Information Theory**, vol. 36, no. 36, pp. 171–173, 1990.

[16] S. V. Ramanan, **Private Communication.**

[17] H. F. Jordan, K. Y. Lee, and D. Lee, "Multichannel Time Slot Permuters," in K. Annamalai, ed., (**High-Speed Fiber Networks and Channels II**), Vol. Proc. SPIE 1874 - paper 26, Sept 8-9, 1992.

[18] D. K. Hunter, I. Andonovic, D. G. Smith, , and B. Culshaw, "Architecture for Optical TDM Switching," **Electronics Letters**, vol. 28, pp. 1746–1747, Aug 1992.

[19] M. G. Hluchyj and M. J. Karol, "ShuffleNet: An application of Generalized Perfect Shuffles to Multihop Lightwave Networks," in **Proceedings of IEEE INFOCOM '88**, IEEE Computer Society Press, Mar 1988, pp. 379–390.

[20] A. S. Acampora and S. A. Shah, "Multihop lightwave networks: A comparison of store-and-forward and hot-potato routing," **IEEE Transactions on Communications**, vol. 40, pp. 1082–1090, june 1992.

[21] J. Bannister, F. Borgonovo, L. Fratta, and M. Gerla, "A versatile model for predicting the performance of deflection-routing networks," **Performance Evaluation**, vol. 16, pp. 201–222, 1992.

[22] Z. Zhang and A. S. Acampora, "Analisis of multihop lightwave networks," in **Proceedings of Globecom '90 Conference**, December 1990, pp. 1873–1878.

[23] J. T. Schwartz, "Ultracomputers," **ACM TOPLAS**, vol. 2, pp. 484–521, 1980.

[24] G. E. Schmidt, "The butterfly parallel processor," in **Proceedings of the Second International Conference on Parallel Processing**, 1987, pp. 362–365.

[25] R. J. Swan, A. Bechtolsheim, K. W. Lai, and J. K. Ousterhout, "The implementation of the Cm* multi-micpproprocessor," in **Proceedings of the National Computer Conference**, 1977, pp. 645–654.

[26] D. D. Gajski, D. J. Kuck, D. H. Lawrie, and A. H. Sameh, "Cedar: A large scale multiprocessor," in **Proceedings of the 1983 International Conference on Parallel Processing**, Aug. 1983, pp. 524–529.

[27] J. T. Kuehn and B. J. Smith, "The Horizon supercomputing system: Architecture and software," in **supercomputing '88**, 1988, pp. 28–34.

[28] A. Agarwal, D. Chaikan, K. Johnson, D. Kranz, J. Kubiatowicz, K. Kurihara, B. Lim, g. Maa, and D. Nussbaum, "The mit alewife machine: A large-scale distributed -memory multiprocessor," in M. Dubois and S. Thakkar, eds., (**Scalable Shared Memory Multiprocessors**), pp. 219–261. Kluwer Academic Publishers, 1992.

[29] G. F. Pfister, W. C. Brantley, D. A. George, S. L. Harvey, W. J. Kleinfelder, K. P. McAuliffe, E. A. Melton, V. A. Norton, and J. Weiss, "The IBM research parallel processor prototype(RP3): Introduction and architecture," in **Proceedings of the 1985 International Conference on Parallel Processing**, 1985, pp. 764–771.

[30] B. J. Smith, "A pipelined, shared resource mimd computer," in **1978 International Conference on Parallel Processing**, 1978, pp. 6–8.

[31] A. L. Decegama, **Parallel Processing Architectures and VLSI Hardware, Volume 1.** Englewood Cliffs, New Jersey 07632: Prentice Hall, 1989.

[32] D. Callahan and B. Smith, "A future-based parallel language for a general-purpose highly-parallel computer," in **Languages and Compilers for Parallel Computing**, pp. 95–113. The MIT Press, Cambrige, Mass, 1990.

[33] J. R. H. Halstead and T. Fujita, "Masa: A multithreaded processor architecture for parallel symbolic computing," in **The 15th Annual international Symposium on Computer Architecture**, 1988, pp. 443–451.

[34] R. A. Iannucci, "Towards a dataflow/von neumann hybrid architecture," in **The 15th Annual International Symposium on Computer Architecture**, 1988, pp. 131–140.

[35] W. J. Kaminsky and E. A. Davidson, "Developing a multiple-instruction-stream single-chip processor," **IEEE Computer**, December 1979.

[36] N. P. Topham, A. Omondi, and R. N. Ibbett, "Context flow: An alternative to conventional pipelined architectures," **The Journal of supercomputing**, vol. 2, no. 2, pp. 29–53, 1988.

[37] A. Agarwal, B. H. Lim, D. Kranz, and J. Kubiatowicz, "APRIL:a processor architecture for multiprocessing," in **The 17th Annual Symposium on COMPUTER ARCHITECTURE**, 1990, pp. 104–114.

[38] H. Hirata, K. Kimura, S. Nagamine, Y. Mochizuki, A. Nishimura, Y. Nakase, and T. Nishizawa, "An elementary processor architectures with simultaneous instruction issuing from multiple threads," in **the 19th Annual International Symposium on Computer Architecture**, May 1992, pp. 136–145.

[39] H. F. Jordan, "Data communication in multiprocessors: Shared and fragmented memory," CSDG 88-1, Computer Systems Design Group, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309-0425, January 1988.

[40] D. Litaize, A. Mzoughi, C. Rochange, and P. Sainrat, "Multithreading: A revisionist view of dataflow architectures," in **the 19th Annual International Symposium on Computer Architecture**, May 1992, pp. 70–79.

[41] K. L. Johnson, "Impact of communication locality on large-scale multiprocessor performance," in **the 19th Annual International Symposium on Computer Architecture**, May 1992, pp. 392–402.

[42] K. Gharachorloo, A. Gupta, and J. Hennessy, "Performance evaluation of memory consistency models for shared-memory multiprocessors," **Sigplan Notices**, vol. 26, Apr 1991.

[43] A. Gupta, J. H. K. Gharachorloo, T. Mowry, and W. Weber, "Comparative evaluation of latency reducing and tolerating techniques," in **The 18th Annual International Symposium on Computer Architecture**, May 1991, pp. 254–263.

[44] B. E. Florenet.al., "Optical interconnects in the touchstone supercomputer system," in **Proceedings SPIE Integrated Optoelectronics for Communication and Processing**, Sept 1991.

[45] A. Wilsonet.al., "Galacticanet:an architecture for shared memory multiprocessing," in **Proceedings 1991 GOMAC**, Nov 1991.

[46] T. A. Laneet.al., "Gigabit optical interconnects for the connection machine," in **Proceedings SPIE Optical Interconnects in the Computer Environment**, Sept 1989.

[47] D. A. B. Miller, "Optics for low energy communication inside digital processors: Quantum detectors, sources, and modulators as efficient impedence converters," **Optics Letters**, vol. Jan., 1989.

[48] L. D. Hutcheson, "High speed optical interconnects for computing applications," in **SPIE Vol.862**, 1987, pp. 2–10.

[49] H. S. Hinton, "Photonics in switching," **IEEE LTS: Special Issue on Gigabit Networks**, pp. 26–35, Aug 1992.

[50] J. W. Goodman, "Optics as an interconnect technology," in **Optical Processing and Computing**, pp. 1–32. Academic Press, 1989.

[51] E. Arthurs, M. S. Goodman, H. Kobrinski, and M. Vecchi, "Hypass: An optoelectronic hybrid packet switching system," **IEEE Journal on Selected Areas in Communications**, vol. 6, pp. 1500–1509, Dec 1988.

[52] H. S. Hinton, "Architectural considerations for photonic switching networks," **IEEE Journal on Selected Areas in Communications**, vol. 6, pp. 1209–1226, Aug 1988.

[53] C. A. Brackett, "Dense wavelength division multiplexing networks: Principles and applications," **IEEE Journal on Selected Areas in Communications**, vol. 8, pp. 948–964, Aug 1990.

[54] "Special issue on dense wavelength division multiplexing techniques for high capacity and multiaccess communication systems," **IEEE Journal on Selected Areas in Communications**, vol. 8, Aug 1990.

[55] P. W. Dowd, "High performance interprocessor communication through optical wavelength division multiple access channels," vol. , pp. 96–105, 1991.

[56] K. N. Sivarajan, "Multihop logical topologies for gigabit lightwave networks," **IEEE LTS: Special Issue on Gigabit Networks**, pp. 20–25, Aug 1992.

[57] J. R. Sauer, D. J. Blumenthal, and A. V. Ramanan, "Photonic Interconnects for Multicomputer Communications," **IEEE LTS: Special Issue on Gigabit Networks**, pp. 12–19, Aug 1992.

[58] D. J. Blumenthal, **Multiwavelength Photonic Packet Switched Interconnects.** PhD thesis, University of Colorado at Boulder, Department of Electrical and Computer Engineering, 1993.

[59] D. H. Lawrie and D. A. Padua, "Analysis of message switching with shuffle-exchanges in multiprocessors," in **The Proceedings of the Workshop on Interconnection Networks for Parallel and Distributed Processing**, 1980, pp. 116–123.

[60] T. G. Robertazzi, "Toriodal Networks," **IEEE Communications Magazine**, vol. 26, pp. 45–50, June 1988.

[61] M. P. Pease, "The indirect binary n-cube microprocessor array," **IEEE Transactions on Computers**, vol. C-26, pp. 458–473, May 1977.

[62] D. J. Blumenthal, K. Y. Chen, J. Ma, R. J. Feuerstein, and J. R. Sauer, "Demonstration of a Deflection Routing 2x2 Photonic Switch for Computer Interconnects," **IEEE Photonics Technology Letters**, vol. 4, no. 4, pp. 169–173, 1992.

[63] D. Blumenthal and J. Sauer, "Multiwavelength information processing in Gigabit photonic switching networks," in **Proceedings of the SPIE, Conference on Multigigabit Fiber Communications**, Vol. 1787, (Boston, MA), Sep. 1992.

[64] A. G. Greenberg and J. Goodman, "Sharp approximate models of deflection routing in mesh networks," 11212-890314-03tm, AT&T Bell Laboratories, March 1989.

[65] A. K. Choudhury and V. O. K. Li, "Performance analisis of deflection routing in the Maanhattan street network," in **Proceedings of ICC '91**, IEEE Computer Society Press, 1991, pp. 1659–1665.

[66] J. Brassil and R. Cruz, "Nonuniform traffic in the Manhattan street network," in **Proceedings of ICC '91**, IEEE Computer Society Press, 1991, pp. 1647–1651.

[67] N. F. Maxemchuk, "Comparison of deflection and store-and-forward techniques in the manhattan street and shuffle-exchange networks," in **Proceedings of IEEE INFOCOM '89, The Conference on Computer Communications**, 1990, pp. 800–809.

[68] T. Robertazzi and A. A. Lazar, "Deflection strategies for the manhattan street network," in **Proceedings of ICC '91**, IEEE Computer Society Press, 1991, pp. 1652–1658.

[69] E. Ayanoglu, "Signal flow graphs for path enumeration and deflection routing analysis in multihop networks," in **Proceedings of Globecom '89 Conference**, (Dallas, TX), IEEE Press, 1989.

[70] P. Schweitzer, "Approximate analysis of multiclass closed networks of queues," in **Proceedings of the International Conference on Stochastic Control and Optimization**, 1979.

[71] M. Reiser and S. Lavenberg, "Mean value analysis of closed multichain queuing networks," **JACM**, vol. 27, pp. 313–322, Apr 1980.

[72] P. Schweitzer, "An analytical model of multistage interconnection networks," in **Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Compuer Systems**, May 1990, pp. 192–202.

[73] D. B. Wagner, "Approximate mean value analysis of deflection-routed shuffle-loop networks," CU-CS-608-92, Department of Computer Science, University of Colorado, Boulder, CO 80309-0430, Revised Sep 1992.

[74] I. Chlamtac and A. Fumagalli, "An All-Optical Switch Architecture for Manhattan Networks," **IEEE Journal on Selected Areas in Communications**, vol. 11, pp. 550–559, May 1993.

[75] I. Chlamtac and A. Fumagalli, "QUADRO-Star: High Performance Optical WDM Star Networks," in **IEEE GLOBECOM '91**, Dec 1991, pp. 1224–1229.

[76] I. Chlamtac and A. Fumagalli, "All Optical Double Ring Network," in **Third Workshop on Very High Speed Networks**, March 1992.

[77] F. Pittelli and D. Smitley, "Analysis of a 3D toroidal network [illegible] a shared memory architecure," in **supercomputing '88**, 1988, pp. 42–47.

[78] H. F. Jordan, "HEP architecture, programming and performance," in **Parallel MIMD Computation: HEP Supercomputer and Its Applications**, pp. 1–40. The MIT Press, 1985.

[79] J. R. Sauer, "Multi-gbit/s optical computer interconnect," in **Proceedings of the SPIE**, sep. 1990.

[80] J. T. Brassil and R. L. Cruz, "An Approximate Analysis of Packet Transit Delay in the Manhattan Street Network," **IEEE Transactions on Communications**, submitted Feb 1990.

[81] Z. Zhang and A. S. Acampora, "Performance analysis of multihop lightwave networks with hot-potato routing and distance-age priorities," in **Proceedings of IEEE INFOCOM '91**, April 1991, pp. 1012–1021.

[82] R. Jain, **The Art of Computer Systems Performance Analysis**. John Wiley & Sons, Inc., 1991.

[83] N. F. Maxemchuk, "Routing in the Manhattan street network," **IEEE Transactions on Communications**, vol. COM-35, pp. 503–512, May 1987.

[84] S. V. Ramanan and H. F. Jordan, "Photonic Architectures for performing Perfect Shuffle on a Time-Division multiplexed signal," in J. Pazaris and G. W. eds., eds., (**Optical Inerconnects in the Computer Environment**), Vol. Proc. SPIE 1178, Sept 6-7, 1989.

[85] S. V. Ramanan and H. F. Jordan, "Serial array Shuffle-Exchange architecture for universal permutation of time-slots," in R. Arrathoon, ed., (**Digital Optical Computing II**), Vol. Proc. SPIE 1215, Jan 17-19, 1990.

[86] R. A. Thompson and P. Giardano, "An Experimental Photonic Time-Slot Interchanger Using Optical Fibers as Reentrant Deay-Line Memories,"

Journal of Lightwave Technology, vol. Jan., 1987.

[87] A. K. Nanda, H. Shing, T. Tzen, and L. M. Ni, "resource contention in shared memory multiprocessors: A parameterized performance degradation model," **Journal of Parralel and Distributed Computing**, vol. 12, no. 12, pp. 313–328, 1991.

[88] G. F. Pfister and V. A. Norton, "Hot SpotContention and Combining in Multistage Interconnection Networks," in **Proceedings of the 1985 International Conference on Parallel Processing**, 1985, pp. 790–797.

[89] P-C Yew and N-F Tzeng and D. H. Lawrie, "Disributing Hot-Spot Addressing in Large Sacle Multiprocessors," in **Proceedings of the 1986 International Conference on Parallel Processing**, 1986, pp. 51–58.

[90] M. Kumar and G. F. Pfister, "The Onset of Hot-Spot Contention," in **Proceedings of the 1986 International Conference on Parallel Processing**, 1986, pp. 28–34.

[91] S. P. Dandamudi and D. L. Eager, "Hot-Spot Contention in Binary Hypercube Networks," **IEEE Transactions on Computers**, vol. C-41, pp. 239–244, Feb 1992.

[92] D. H. Lawrie, "Alignment and access of data in an array processor," **IEEE Transactions on Computers**, vol. C-24, pp. 1145–1155, Dec 1975.

[93] D. M. Dias and J. R. Jump, "Ananysis and Simulation of Buffered Delta Networks," **IEEE Transactions on Computers**, vol. C-30, pp. 273–282, April 1981.